Dimensionality Reduction

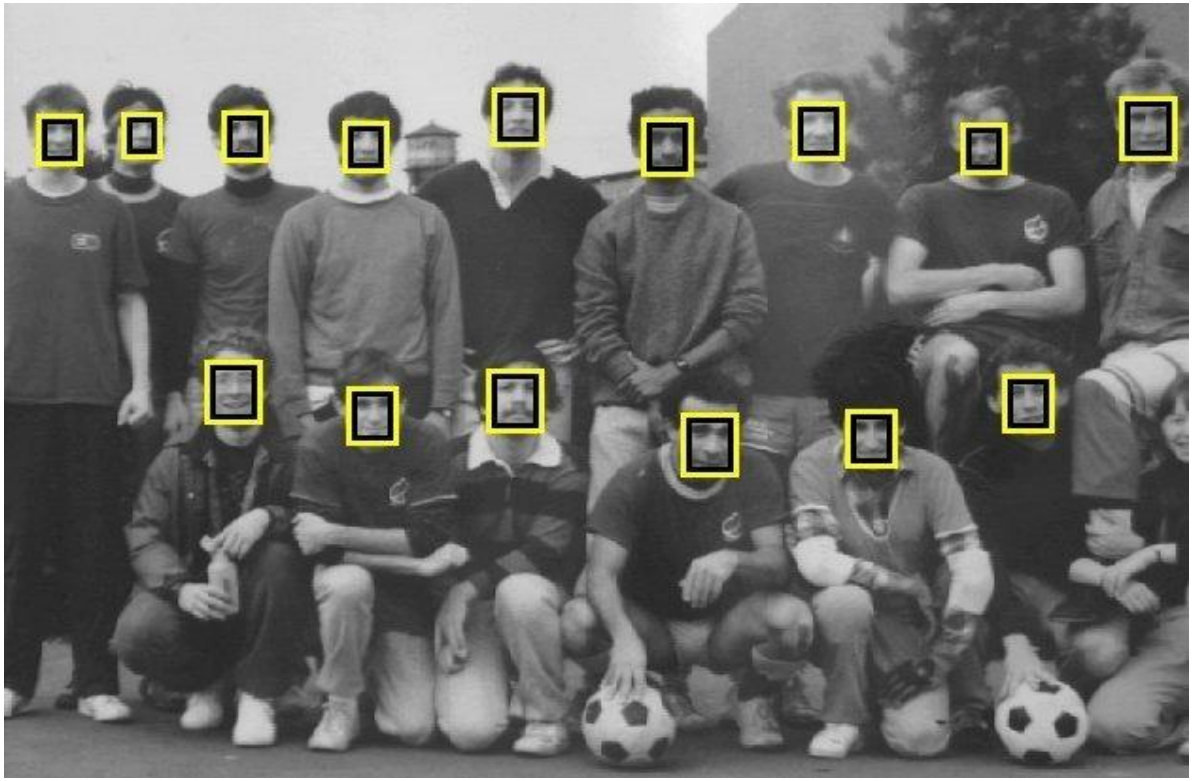
MAN-522: Computer Vision

Data Processing and Representation

Principal Component Analysis (PCA)

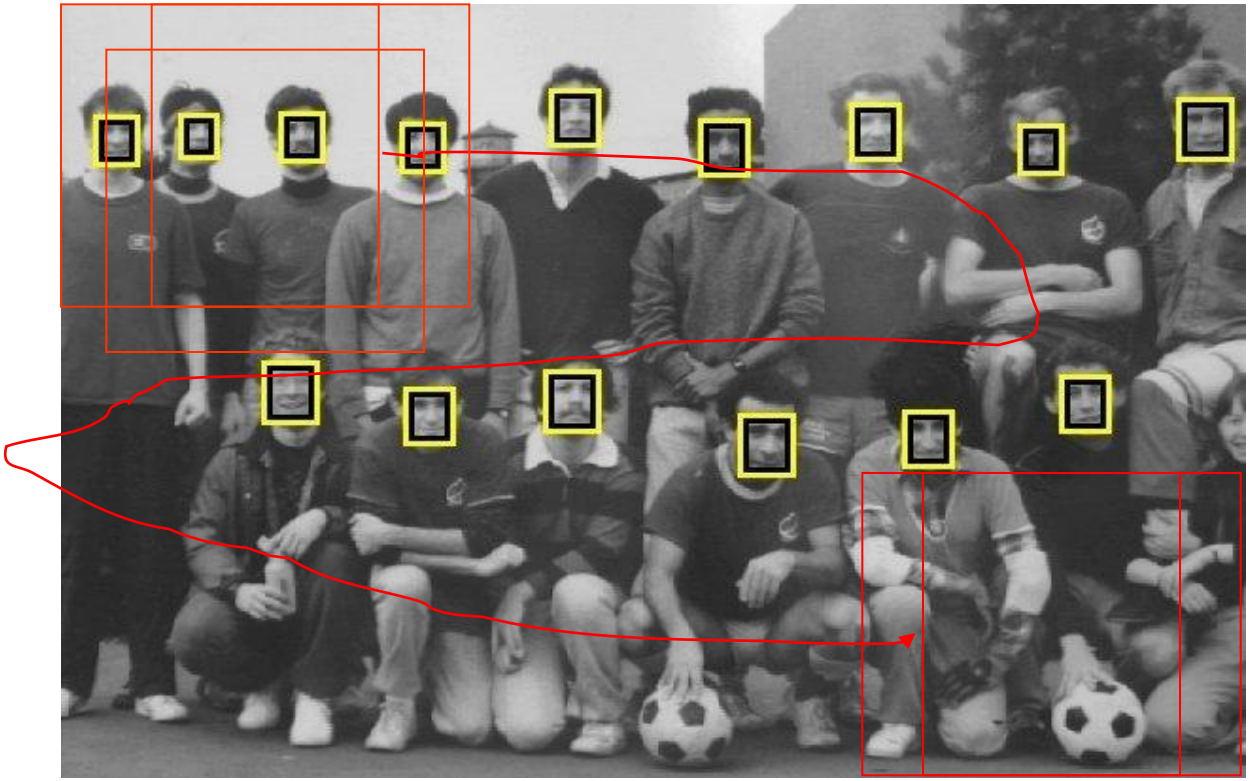
Problems

- Object Detection



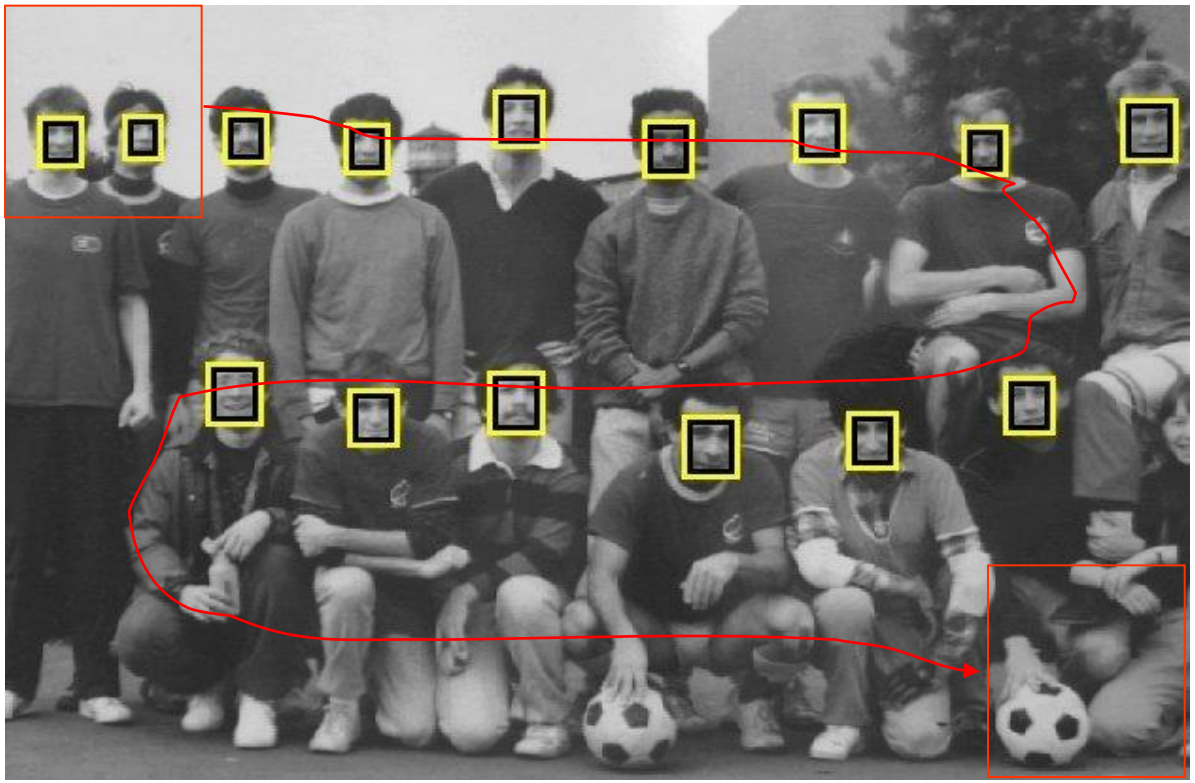
Problems

- Object Detection: Many detection windows



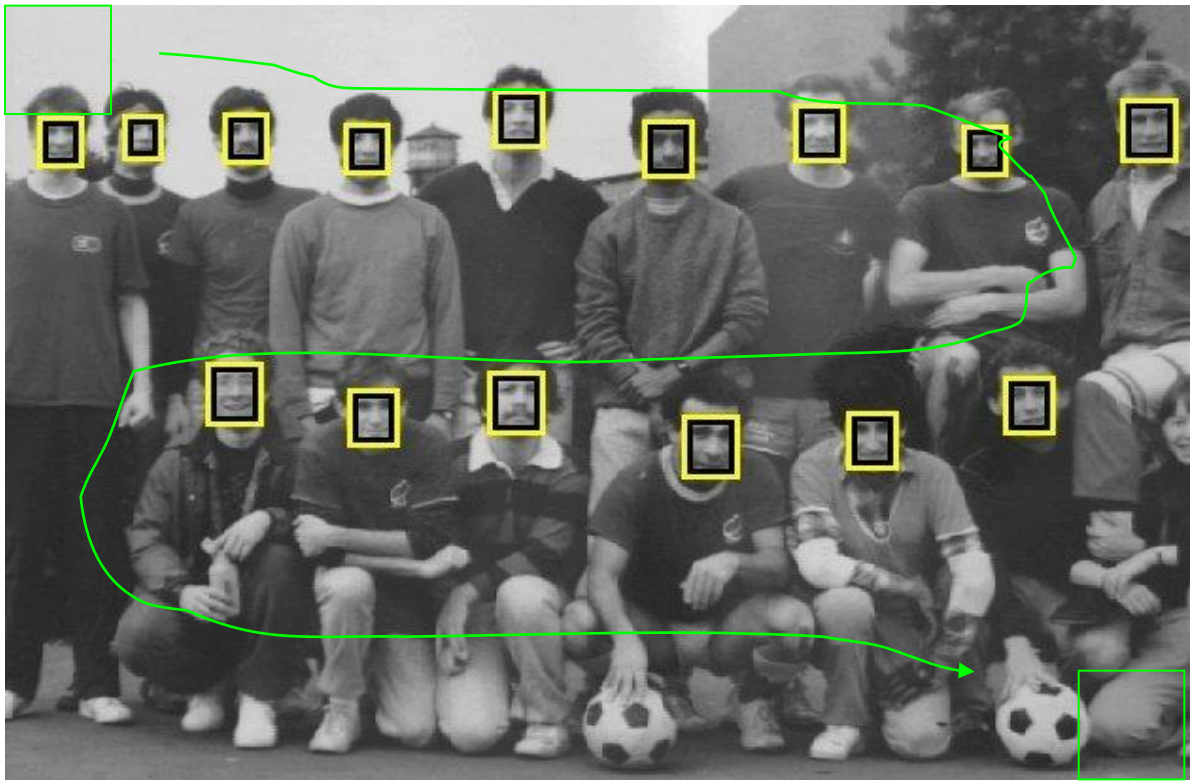
Problems

- Object Detection: Many detection windows



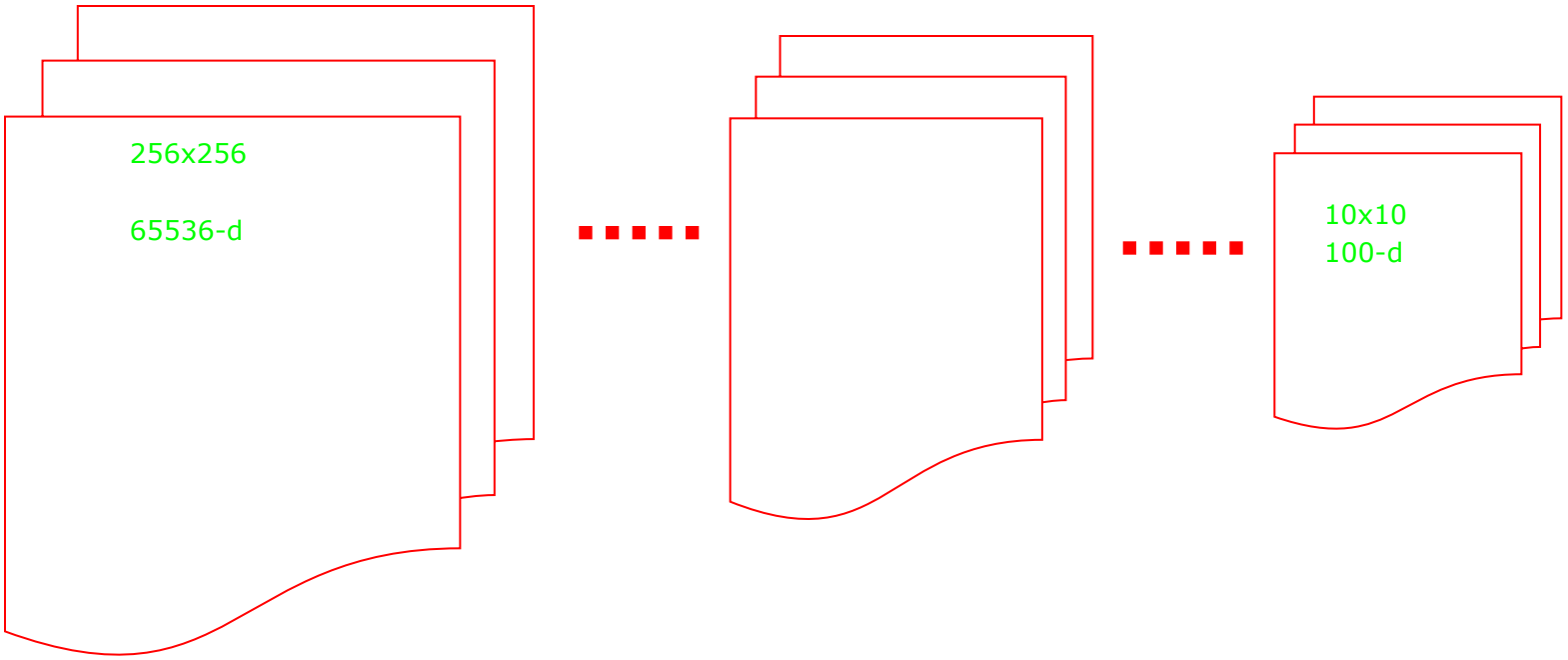
Problems

- Object Detection: Many detection windows



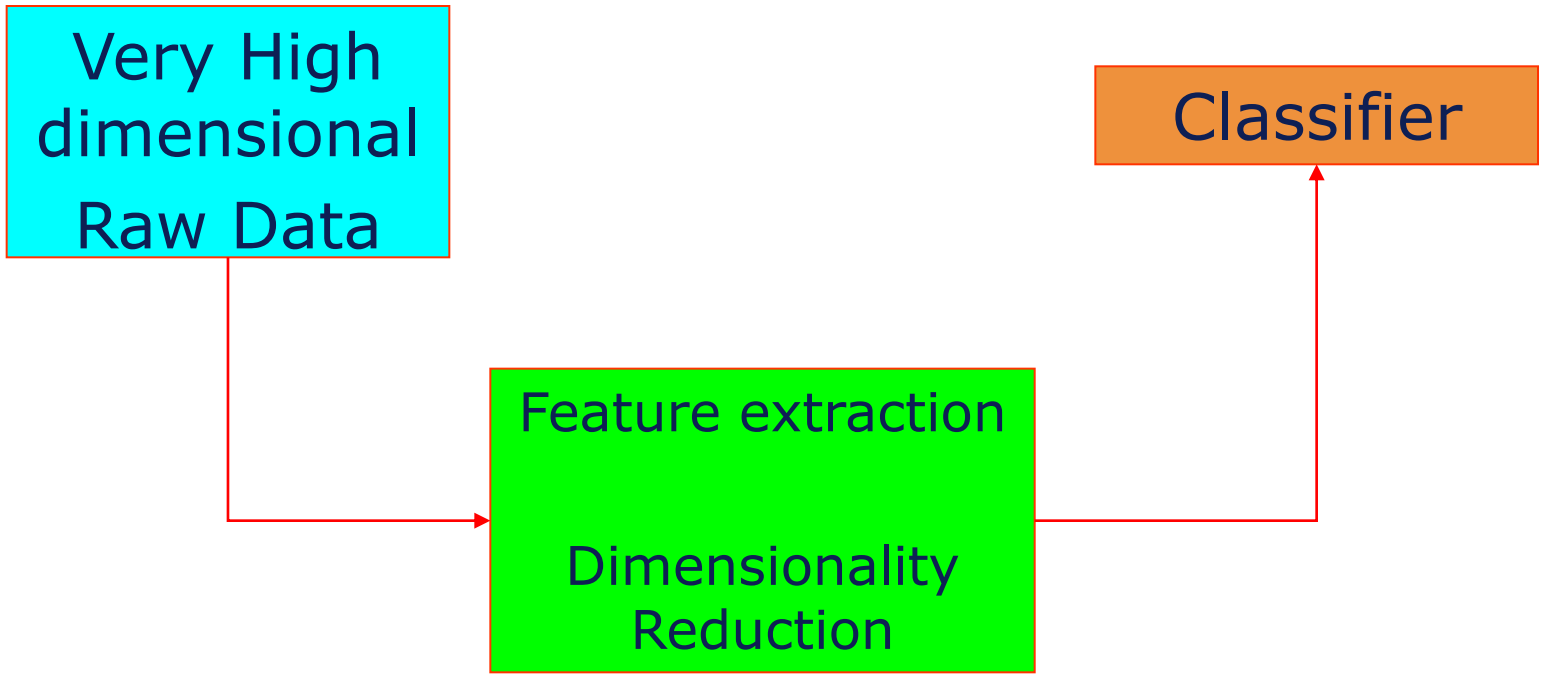
Problems

- Object Detection: Each window is very high dimension data



Processing Methods

- General framework



Feature extraction/Dimensionality reduction

- It is impossible to processing raw image data (pixels) directly
  - Too many of them (or data dimensionality too high)
  - Curse of dimensionality problem
- Process the raw pixel to produce a smaller set of numbers which will capture most information contained in the original data – this is often called a feature vector

Feature extraction/Dimensionality reduction

- Basic Principle
  - From a raw data (vector) X of N-dimension to a new vector Y of n-dimensional (n < < N) via a transformation matrix A such that Y will capture most information in X

$$Y = AX = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} a_{11} & \cdots & a_{1N} \\ & \cdots & \\ \cdots & \cdots & \cdots \quad \cdots \\ a_{n1} & \cdots & a_{nN} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \\ \vdots \\ \vdots \\ \\ x_N \end{bmatrix}$$

PCA

- Principal Component Analysis (PCA) is one of the most often used dimensionality reduction technique.

PCA Goal

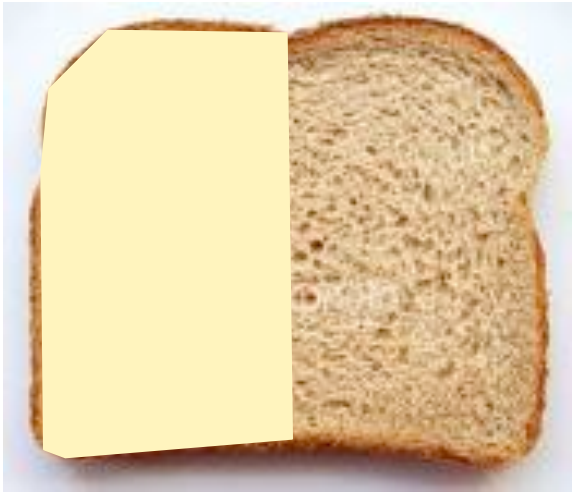
We wish to explain/summarize the underlying variance-covariance structure of a large set of variables through a few linear combinations of these variables.

Applications

- Data Visualization
- Data Reduction
- Data Classification
- Trend Analysis
- Factor Analysis
- Noise Reduction

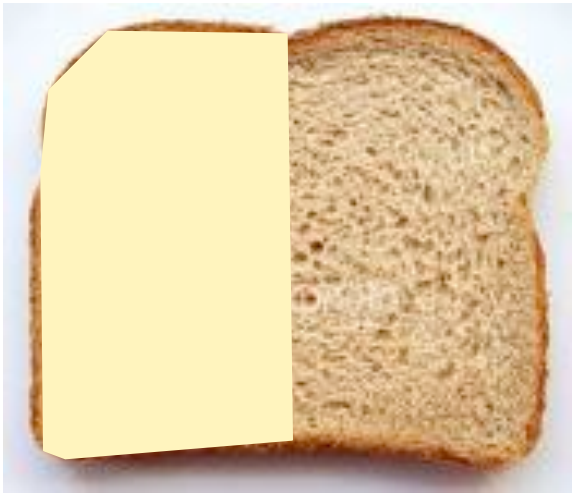
Eigenvectors and Eigenvalues

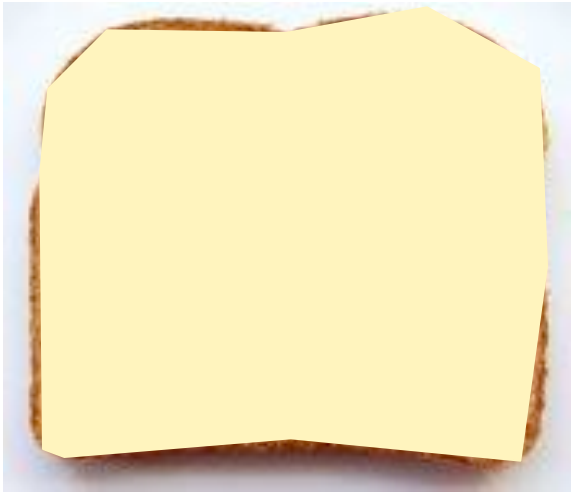
- **Consider this problem of spreading butter on a bread slice**



Eigenvectors and Eigenvalues

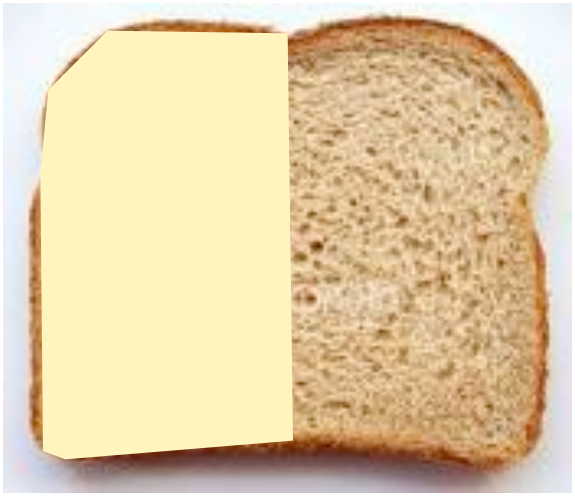
- **Consider this problem of stretching cheese on a bread slice**





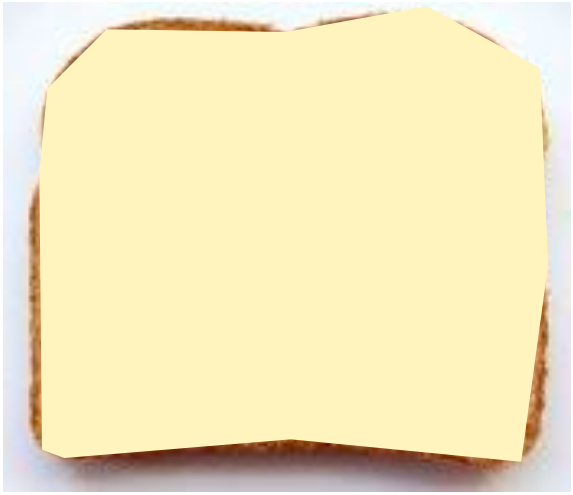
Eigenvectors and Eigenvalues

- **Consider this problem of spreading butter on a bread slice**



$$A = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}$$

Linear Transformation



Eigenvectors and Eigenvalues

Find Eigenvalues:

$$A = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}$$

$$|A - \lambda I| = 0$$

$$\begin{pmatrix} 2-\lambda & 0 \\ 0 & 1-\lambda \end{pmatrix} = 0$$

$$(2-\lambda)(1-\lambda) = 0$$

$$\Rightarrow \lambda = 2,1$$

Eigenvectors and Eigenvalues

Find Eigenvectors for λ=2

$$A\vec{v} = \lambda\vec{v}$$

$$(A - \lambda I)\vec{v} = 0$$

$$\begin{pmatrix} 2-2 & 0 \\ 0 & 1-2 \end{pmatrix}\begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = 0$$

$$0v_1 - 1v_2 = 0$$

$$v_2 = 0$$

$$\vec{v} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

X-axis

Eigenvectors and Eigenvalues

Find Eigenvectors for λ=2

$$A\vec{v} = \lambda\vec{v}$$

$$(A - \lambda I)\vec{v} = 0$$

$$\begin{pmatrix} 2-2 & 0 \\ 0 & 1-2 \end{pmatrix}\begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = 0$$

$$0v_1 - 1v_2 = 0$$

$$v_2 = 0$$

$$\vec{v} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

X-axis

Find Eigenvectors for λ=1

$$A\vec{v} = \lambda\vec{v}$$

$$(A - \lambda I)\vec{v} = 0$$

$$\begin{pmatrix} 2-1 & 0 \\ 0 & 1-1 \end{pmatrix}\begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = 0$$

$$1v_1 - 0v_2 = 0$$

$$v_1 = 0$$

$$\vec{v} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Y-axis

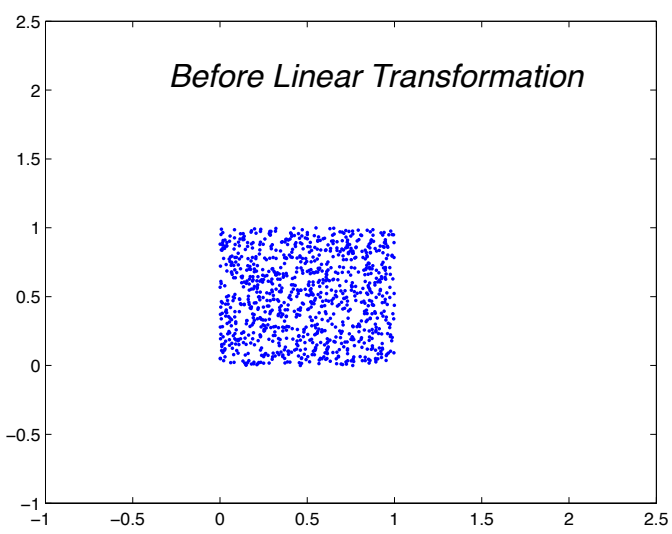
Eigenvectors and Eigenvalues

- **What does the eigenvalue and vectors indicate?**
  - That the linear transformation results in a scaling of λ along the eigenvector corresponding to λ
    - In the example presented, λ=2 along the x-axis and λ=1 along the y-axis (intuitive!)
    - Stated differently, eigenvalue indicates the percentage of transformation present along a particular direction
      - 66.66% along x-axis
      - 33.33% along y-axis

Eigenvectors and Eigenvalues

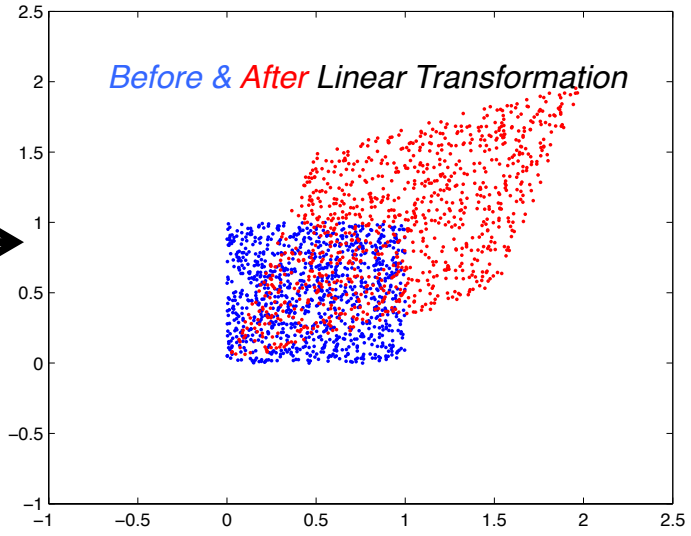
- **Lets consider another example**
  - This time non-zero diagonal elements

$$A = \begin{pmatrix} 1.5 & 0.5 \\ 0.5 & 1.5 \end{pmatrix}$$



$$A = \begin{pmatrix} 1.5 & 0.5 \\ 0.5 & 1.5 \end{pmatrix}$$

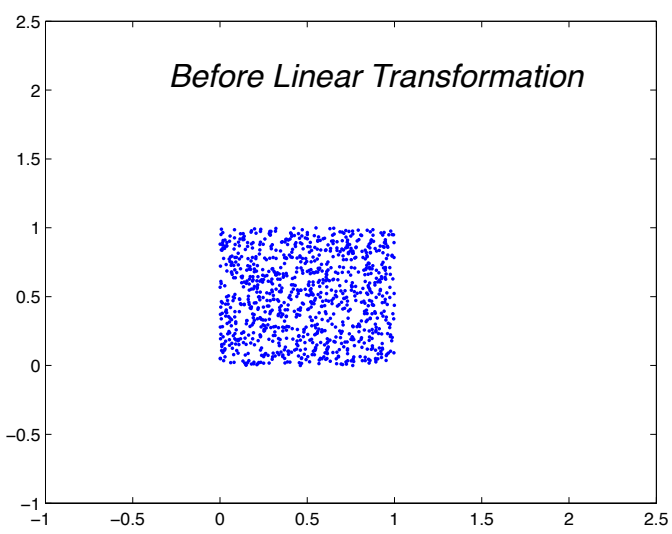
Linear Transformation



Eigenvectors and Eigenvalues

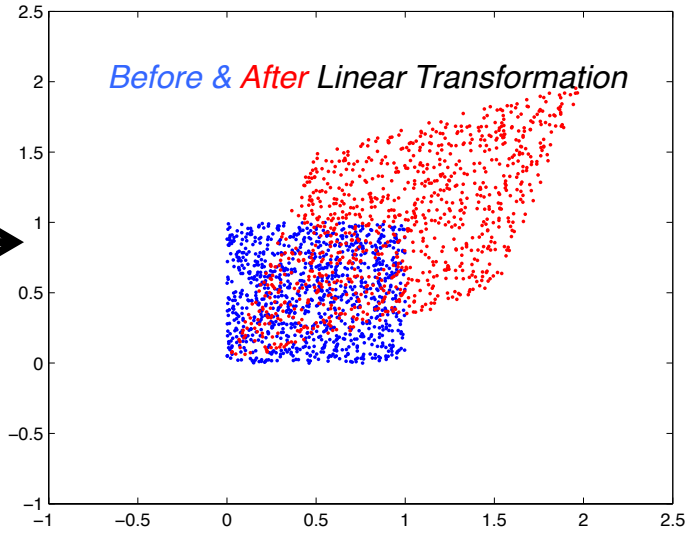
- **Compute eigenvalues and eigenvectors of A**

$$A = \begin{pmatrix} 1.5 & 0.5 \\ 0.5 & 1.5 \end{pmatrix}$$



$$A = \begin{pmatrix} 1.5 & 0.5 \\ 0.5 & 1.5 \end{pmatrix}$$

Linear Transformation



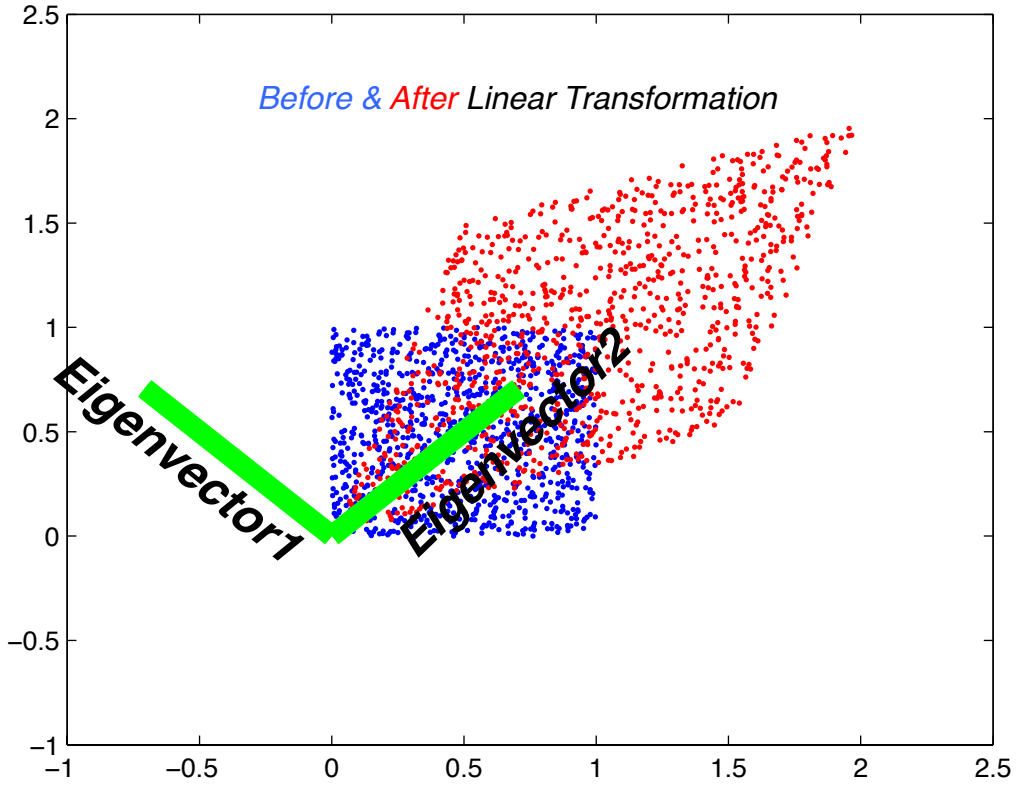
Eigenvectors and Eigenvalues

- **Compute eigenvalues and eigenvectors of A**

$$A = \begin{pmatrix} 1.5 & 0.5 \\ 0.5 & 1.5 \end{pmatrix}$$

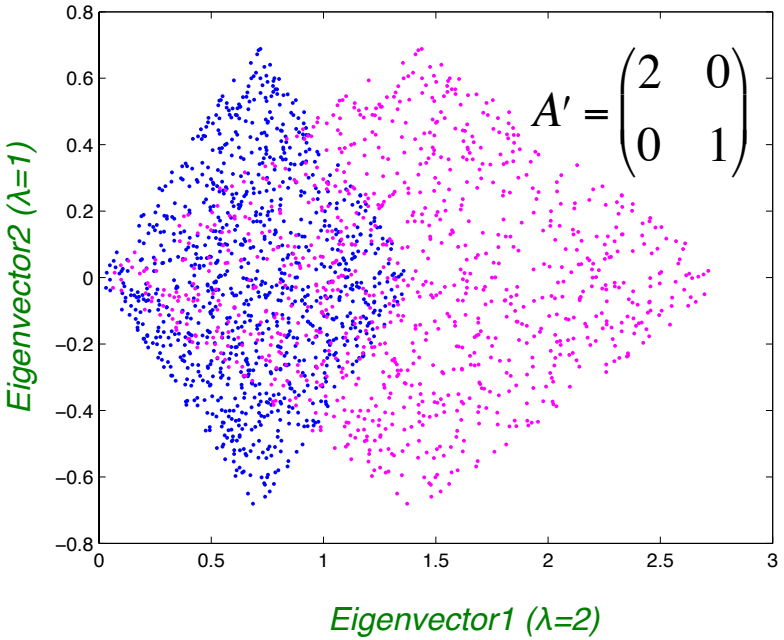
$$\lambda = 1 \qquad \vec{v} = \begin{pmatrix} -0.7071 \\ 0.7071 \end{pmatrix}$$

$$\lambda = 2 \qquad \vec{v} = \begin{pmatrix} 0.7071 \\ 0.7071 \end{pmatrix}$$



Eigenvectors and Eigenvalues

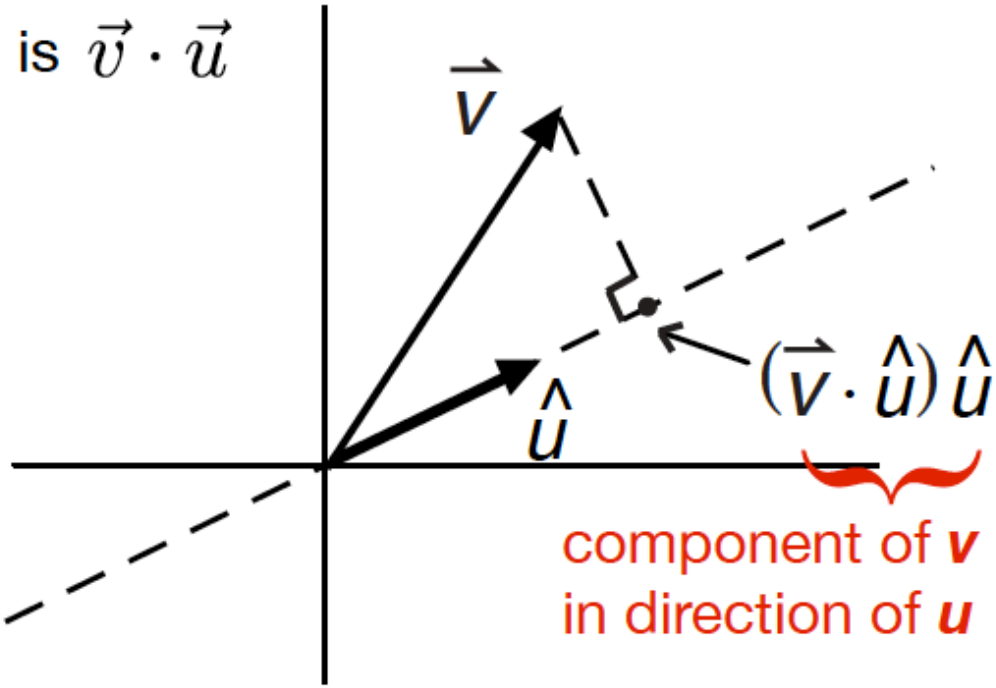
- **What does the eigenvalue and vectors indicate?**
  - That the linear transformation results in a scaling of λ along the eigenvector corresponding to λ
    - λ=2 along $[0.7071, 0.7071]^T$
    - λ=1 along $[-0.7071, 0.7071]^T$



Even though we started with a non-diagonal transformation matrix (A), by computing the eigenvectors and projecting the data onto those eigenvectors allows us to diagonalize the transformation

Recap Linear Algebra: Projection

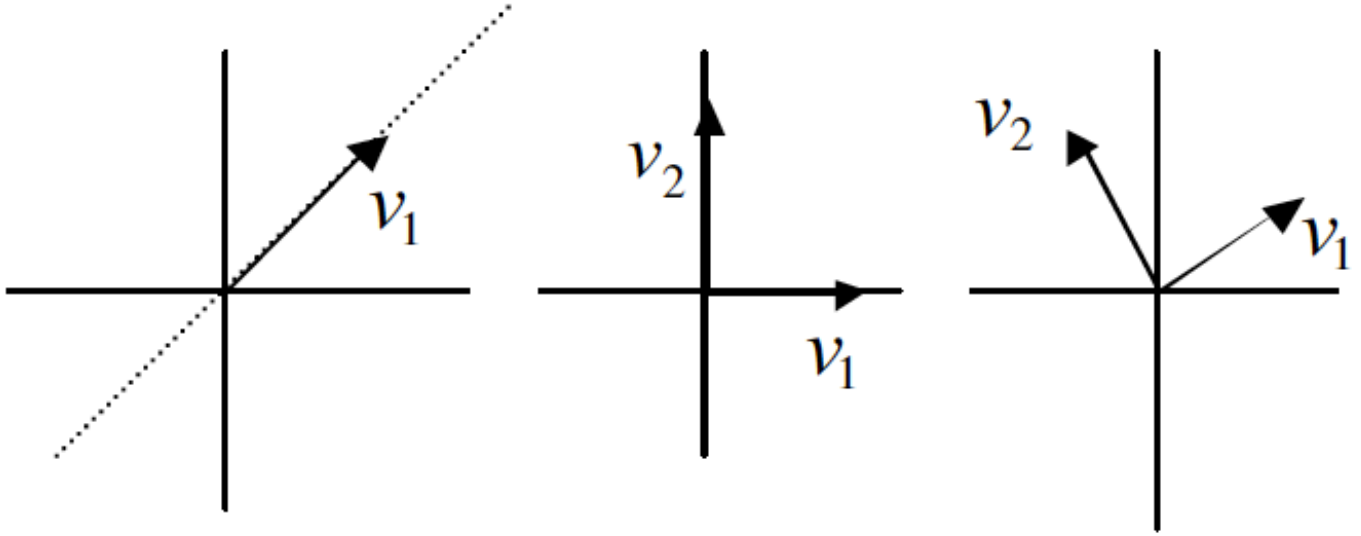
- intuitively, dropping a vector down onto a linear surface at a right angle
- if *u* is a unit vector, length of projection is $\vec{v} \cdot \vec{u}$



- for non-unit vector, length of projection = $\vec{v} \cdot \left(\frac{1}{||\vec{u}||} \vec{u} \right)$

Vector Space

- set of all points that can be obtained by linear combinations of some set of "basis" vectors

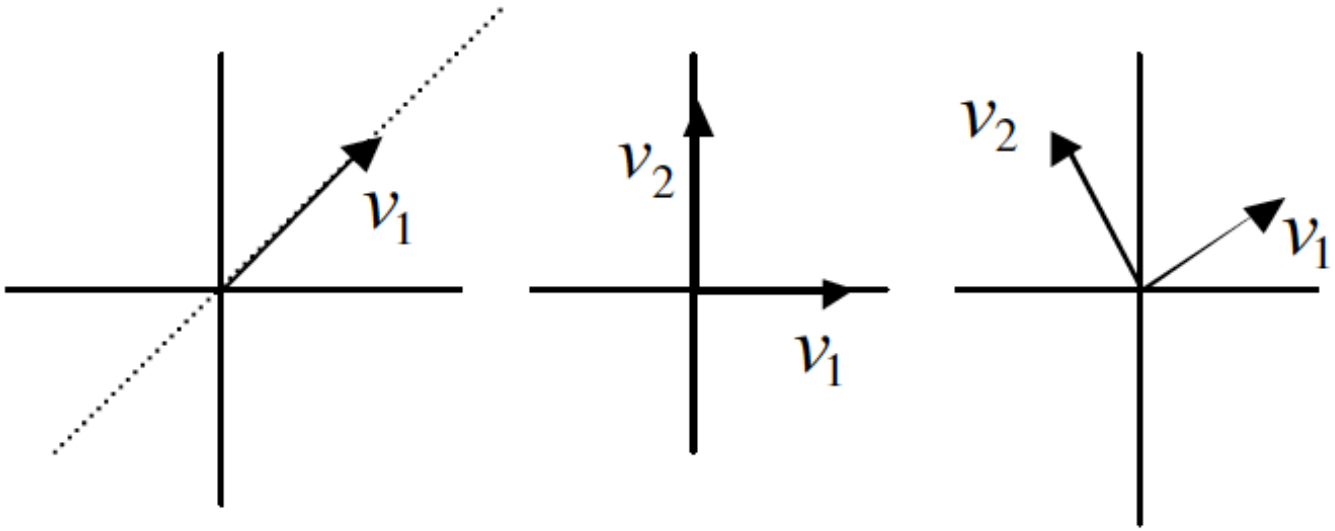


1D vector space spanned by single basis vector

Two different (orthonormal) bases for the same 2D vector space

Basis Vectors

- set of vectors that can "span" (form via linear combination) all points in a vector space



1D vector space
(*subspace of R^2*)

Two different (orthonormal) bases for the same 2D vector space

Projection using Basis Vectors

- Let **B** denote a matrix whose columns form an orthonormal basis for a vector space **W**

$$B = \begin{pmatrix} | & | & & | \\ \vec{b}_1 & \vec{b}_2 & \cdots & \vec{b}_n \\ | & | & & | \end{pmatrix}$$

$$\begin{aligned} \vec{b}_i \cdot \vec{b}_i &= 1 \\ \vec{b}_i \cdot \vec{b}_j &= 0, i \neq j \end{aligned}$$

$$B^T \vec{v} = \begin{pmatrix} \vec{b}_1 \cdot \vec{v} \\ \vdots \\ \vec{b}_n \cdot \vec{v} \end{pmatrix}$$

Vector of projections of ***v*** along each basis vector

Projection using Basis Vectors

- Let **B** denote a matrix whose columns form an orthonormal basis for a vector space **W**

$$B = \begin{pmatrix} | & | & & | \\ \vec{b}_1 & \vec{b}_2 & \cdots & \vec{b}_n \\ | & | & & | \end{pmatrix} \qquad \begin{aligned} \vec{b}_i \cdot \vec{b}_i &= 1 \\ \vec{b}_i \cdot \vec{b}_j &= 0, i \neq j \end{aligned}$$

If **B** is full rank (*n* x *n*), then $BB^T = \left([\vec{b}_i \cdot \vec{b}_j] \right) = I$

$$\Longrightarrow \vec{v} = BB^T \vec{v}$$

we can get back to the original basis through multiplication by B

Projection using Basis Vectors

- In this case (full rank, orthogonal columns), **B** is an *orthogonal matrix*

$$B = \begin{pmatrix} | & | & & | \\ \vec{b}_1 & \vec{b}_2 & \cdots & \vec{b}_n \\ | & | & & | \end{pmatrix} \qquad \begin{aligned} \vec{b}_i \cdot \vec{b}_i &= 1 \\ \vec{b}_i \cdot \vec{b}_j &= 0, i \neq j \end{aligned}$$

Properties:

$$BB^T = B^T B = I$$

$$B^{-1} = B^T$$

$$||B\vec{v}|| = ||B^T\vec{v}|| = ||\vec{v}||$$

length-preserving

Statistics Preliminaries

Mean

- Let $X_1, X_2, , \ldots X_n$ *be n observations of a random variable X*

$$\mu = \overline{X} = E[X] = \frac{1}{n}\sum_{i=1}^{n} X_i$$

- Mean is a measure of central tendency (others are mode and median)

Standard Deviation

- Measure of variability (square root of variance)

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(X_i - \mu)^2}$$

Statistics Preliminaries

Covariance

- A measure of how two variables change together

$$\Sigma_{XY} = \frac{1}{n}\sum_{i=1}^{n}\left(X_i - \mu_X\right)\left(Y_i - \mu_Y\right)^T$$

$$\Sigma_{XY} = E\left[\left(X - E[X]\right)\left(Y - E[Y]\right)\right]$$

Covariance Matrix

$$\Sigma = \begin{pmatrix} \sigma_{X1}^2 & \Sigma_{X1X2} & \cdots & \Sigma_{X1Xd} \\ \Sigma_{X1X2} & \sigma_{X2}^2 & \cdots & \Sigma_{X2Xd} \\ \cdot & & & \cdot \\ \cdot & & \cdots & \cdot \\ \cdot & & & \cdot \\ \Sigma_{X1Xd} & \Sigma_{X2Xd} & \cdots & \sigma_{Xd}^2 \end{pmatrix}$$

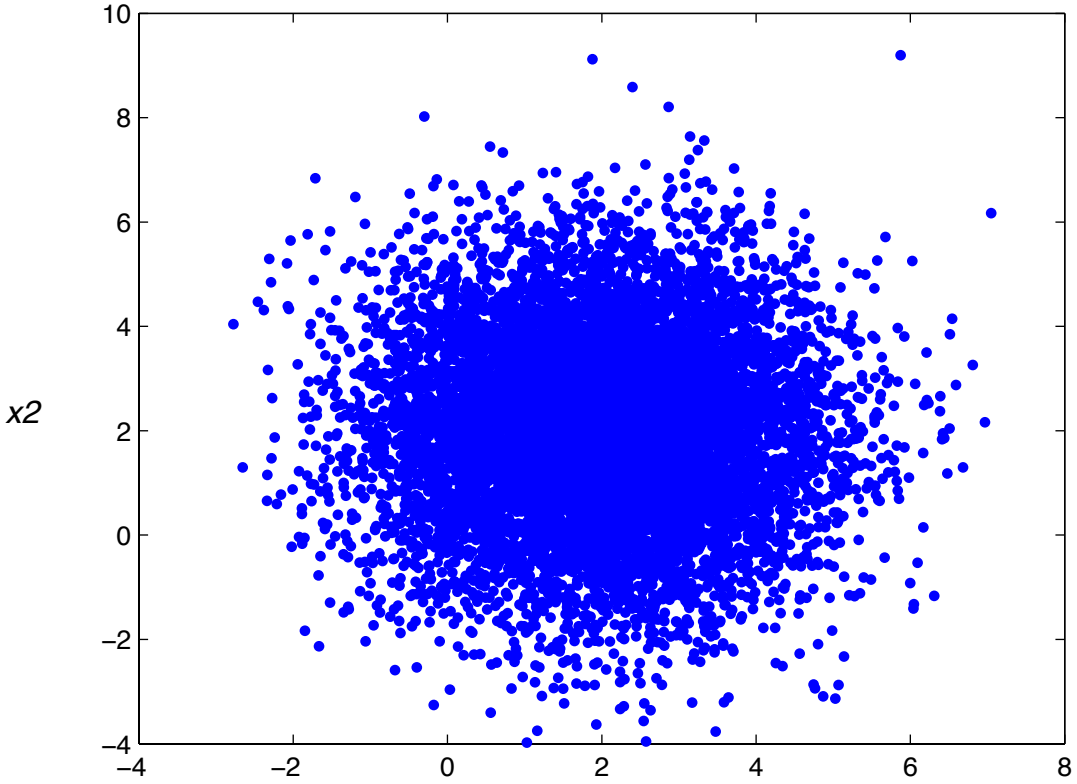
Statistics Preliminaries

- **Correlation**
  - A normalized measure of how two variables change together

$$\rho_{XY} = \frac{\Sigma_{XY}}{\sigma_X \sigma_Y}$$

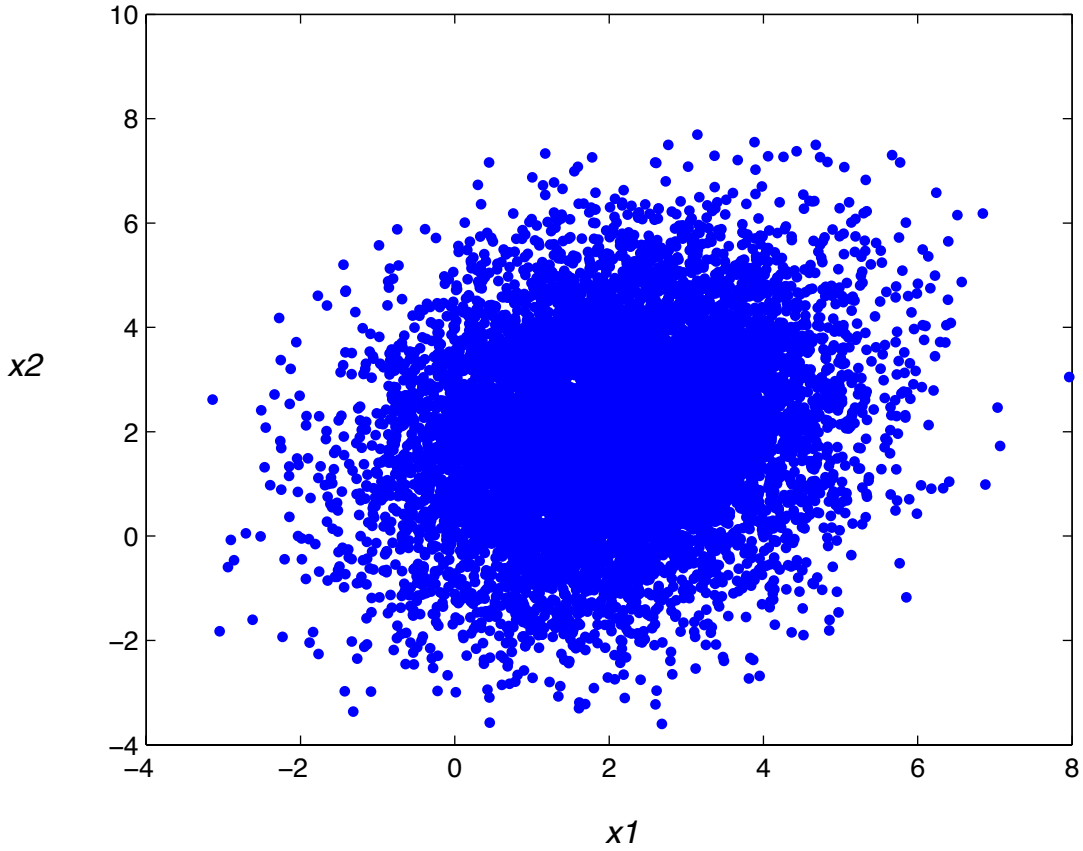
Statistics Preliminaries – An Example

$$\mu = \begin{pmatrix} 2 \\ 3 \end{pmatrix} \qquad \Sigma = \begin{pmatrix} 2 & 0 \\ 0 & 3 \end{pmatrix}$$



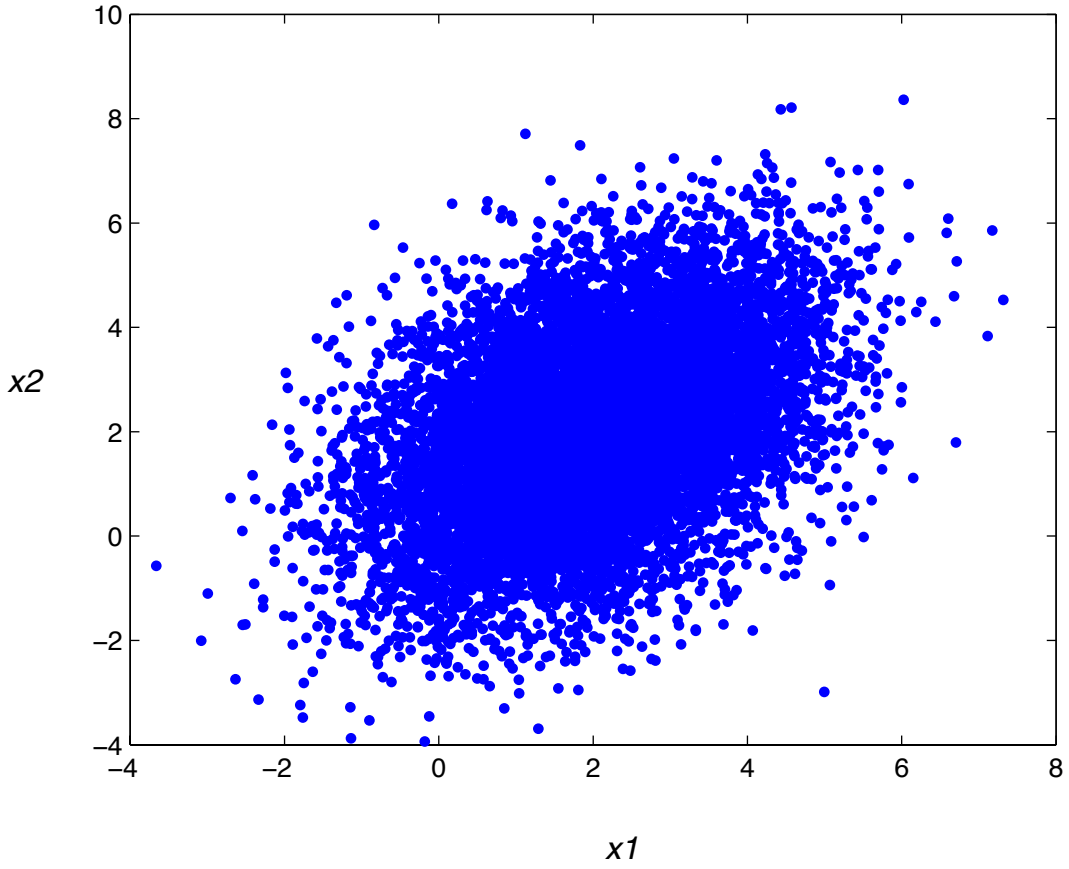
Statistics Preliminaries – An Example

$$\mu = \begin{pmatrix} 2 \\ 3 \end{pmatrix} \qquad \Sigma = \begin{pmatrix} 2 & 0.5 \\ 0.5 & 3 \end{pmatrix}$$



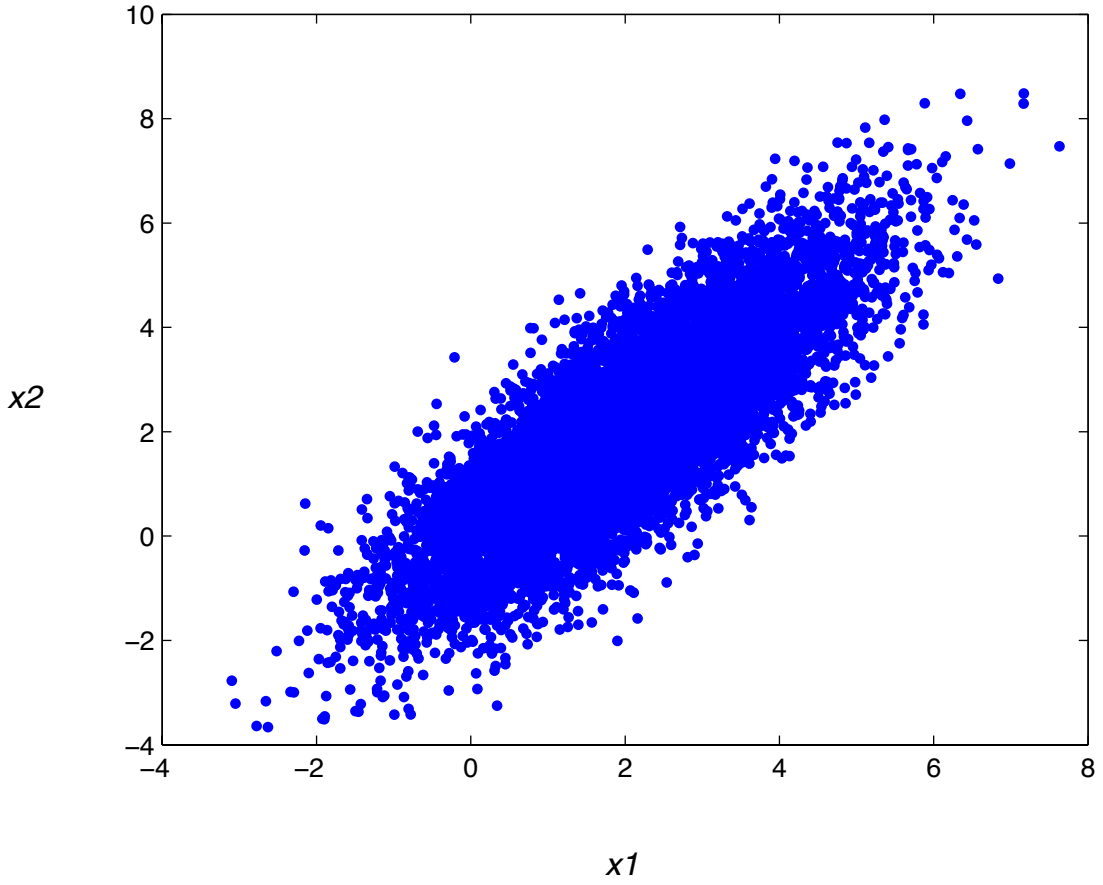
Statistics Preliminaries – An Example

$$\mu = \begin{pmatrix} 2 \\ 3 \end{pmatrix} \qquad \Sigma = \begin{pmatrix} 2 & 1 \\ 1 & 3 \end{pmatrix}$$



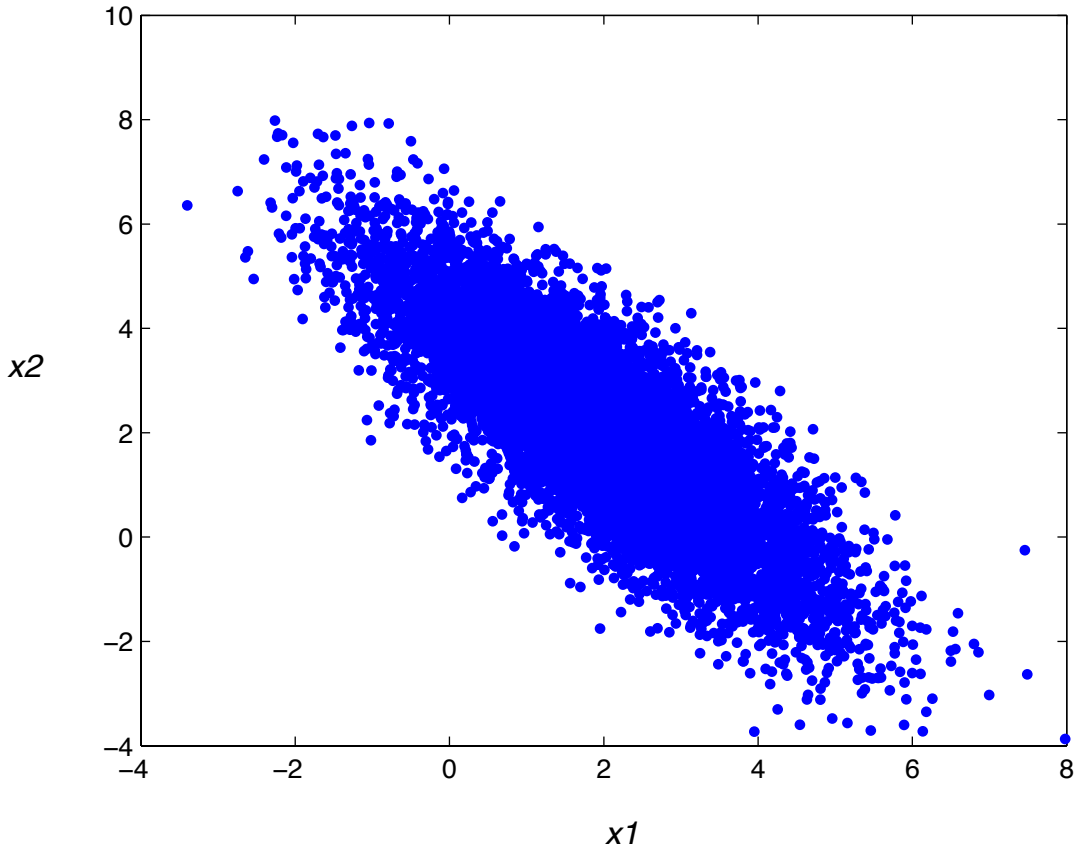
Statistics Preliminaries – An Example

$$\mu = \begin{pmatrix} 2 \\ 3 \end{pmatrix} \qquad \Sigma = \begin{pmatrix} 2 & 2 \\ 2 & 3 \end{pmatrix}$$



Statistics Preliminaries – An Example

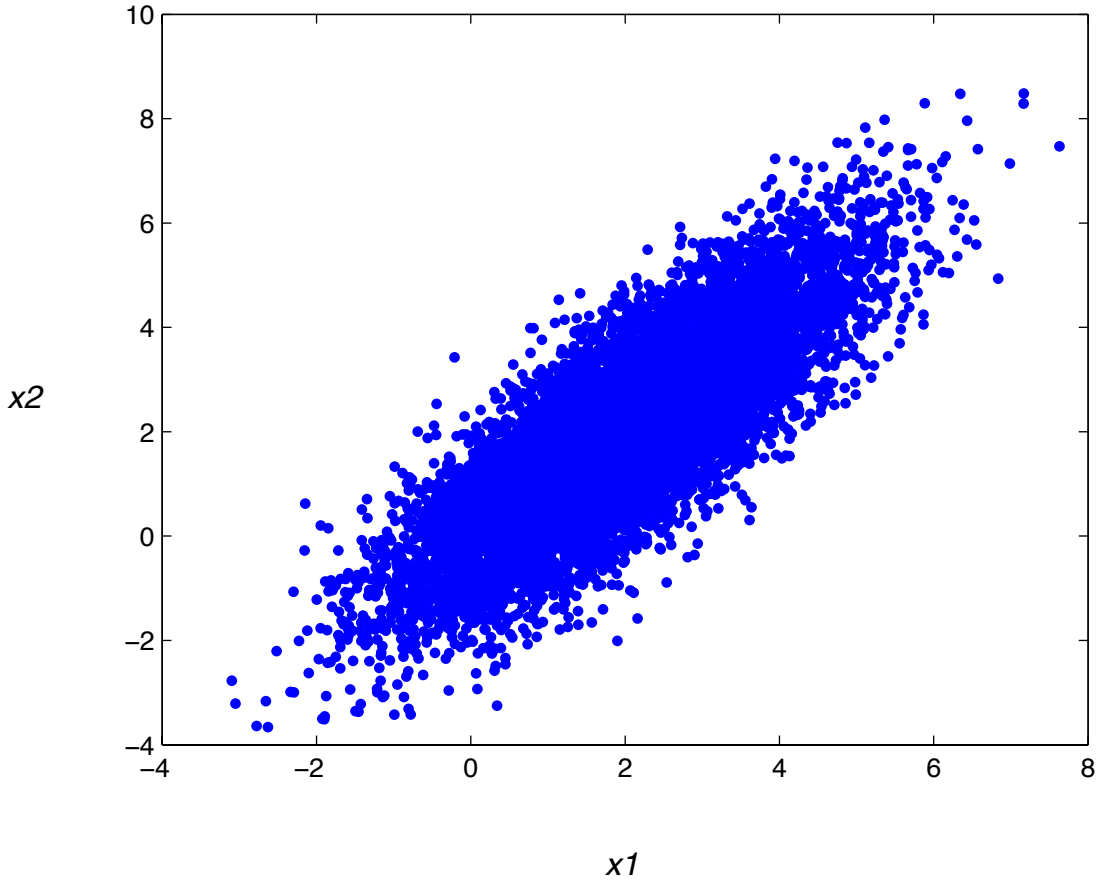
$$\mu = \begin{pmatrix} 2 \\ 3 \end{pmatrix} \qquad \Sigma = \begin{pmatrix} 2 & -2 \\ -2 & 3 \end{pmatrix}$$



PCA – An Example

$$\mu = \begin{pmatrix} 2 \\ 3 \end{pmatrix} \qquad \Sigma = \begin{pmatrix} 2 & 2 \\ 2 & 3 \end{pmatrix}$$

Can you find a vector that would approximate this 2-D space?

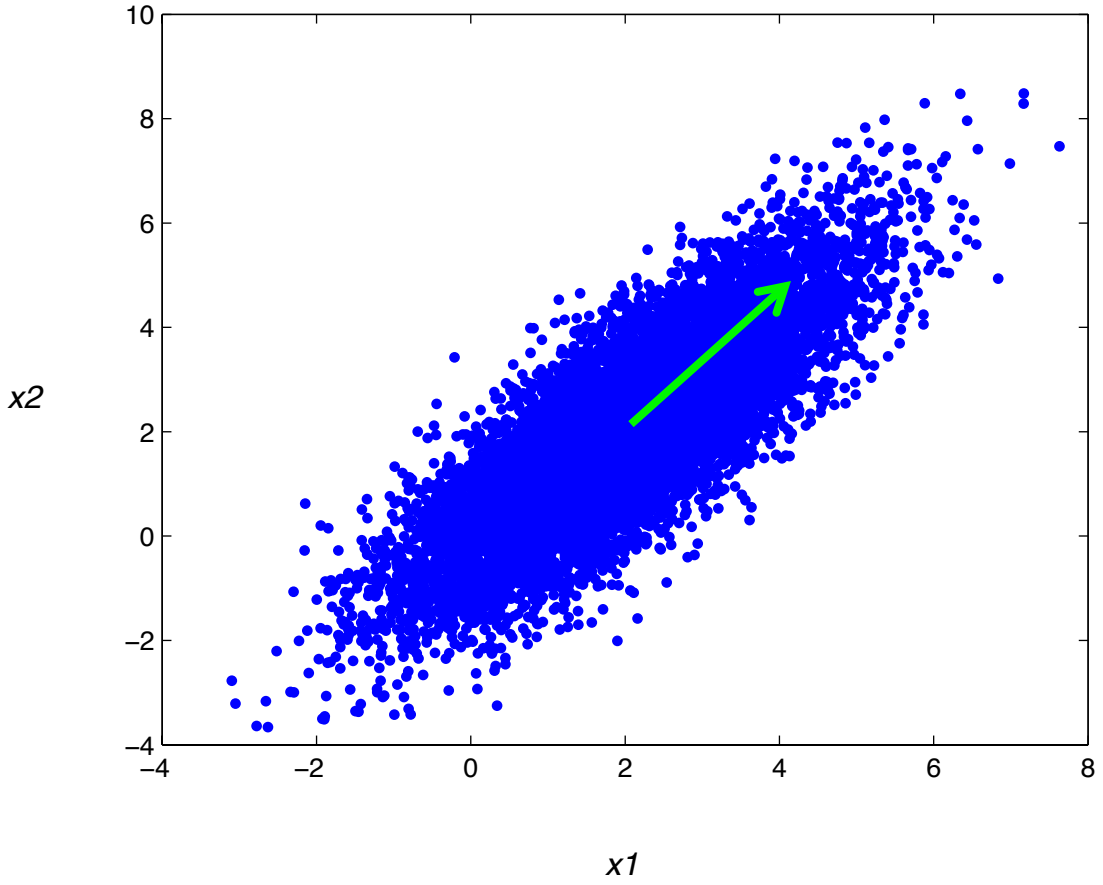


PCA – Let's build some intuition

$$\mu = \begin{pmatrix} 2 \\ 3 \end{pmatrix} \qquad \Sigma = \begin{pmatrix} 2 & 2 \\ 2 & 3 \end{pmatrix}$$

Can you find a vector that would approximate this 2-D space?

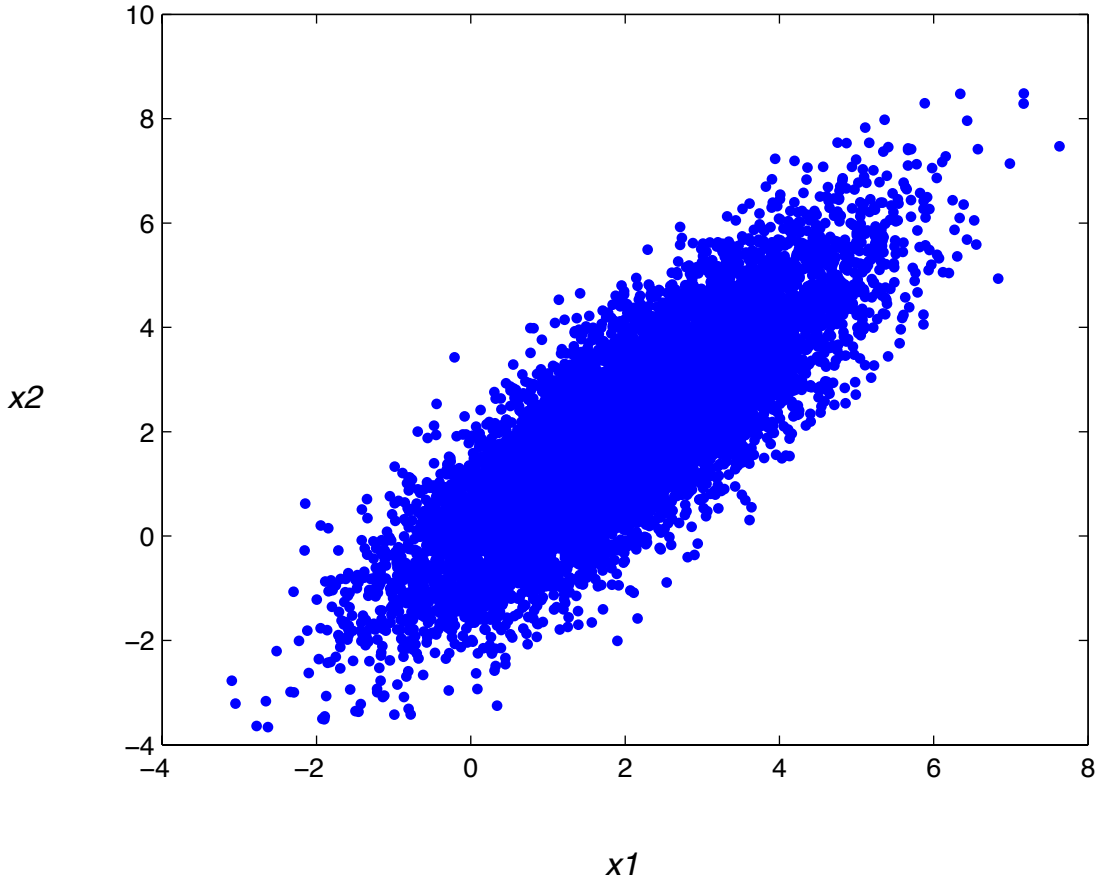
Green vector right?



PCA – Let's build some intuition

$$\mu = \begin{pmatrix} 2 \\ 3 \end{pmatrix} \qquad \Sigma = \begin{pmatrix} 2 & 2 \\ 2 & 3 \end{pmatrix}$$

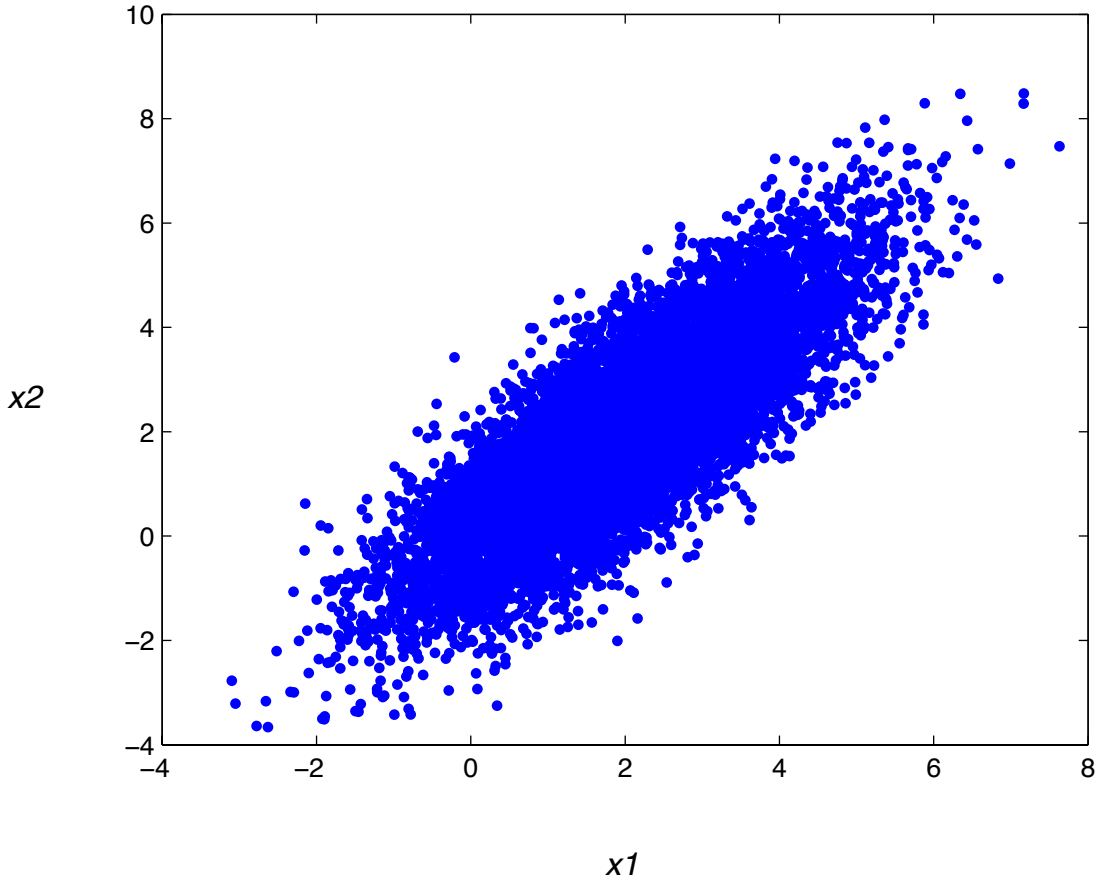
It would be nice to
diagonalize the covariance matrix
then you have only think about variance



PCA – Let's build some intuition

$$\mu = \begin{pmatrix} 2 \\ 3 \end{pmatrix} \qquad \Sigma = \begin{pmatrix} 2 & 2 \\ 2 & 3 \end{pmatrix}$$

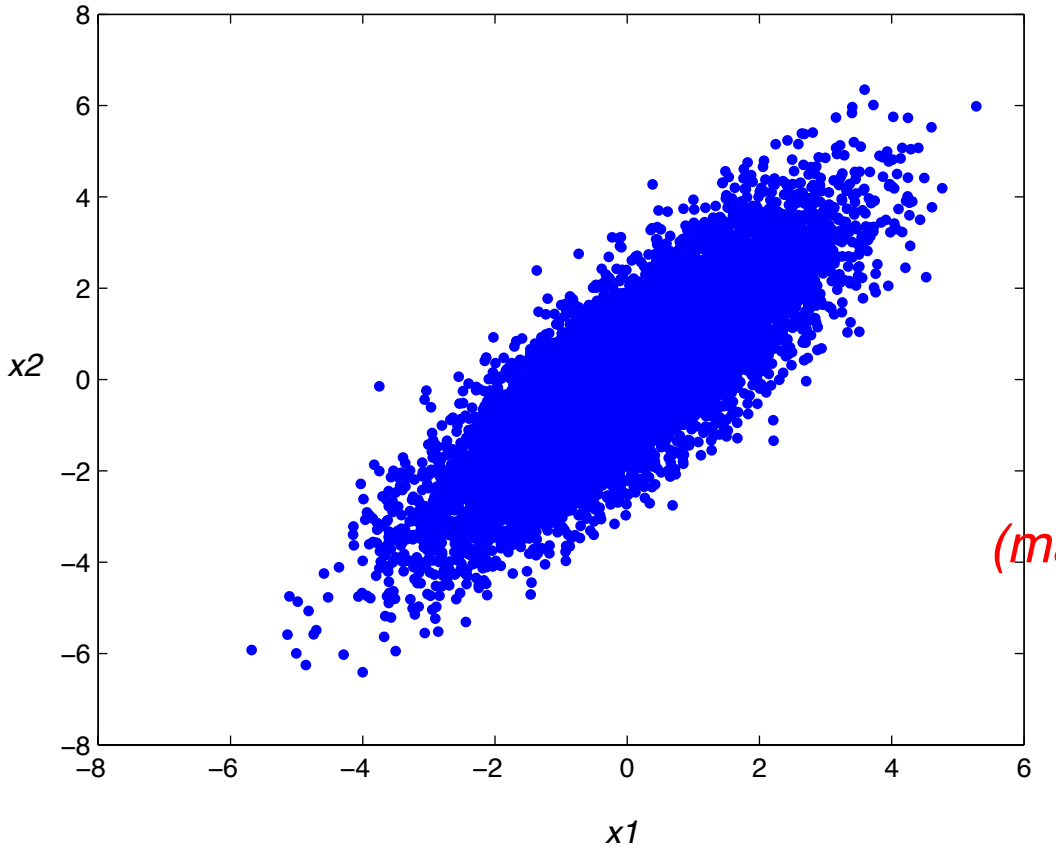
It would be nice to
diagonalize the covariance matrix
then you have only think about variance
Think eigenvectors of covariance matrix



PCA – Let's build some intuition

$$\mu = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \qquad \Sigma = \begin{pmatrix} 2 & 2 \\ 2 & 3 \end{pmatrix}$$

It would be nice to
diagonalize the covariance matrix
then you have only think about variance
Think eigenvectors of covariance matrix

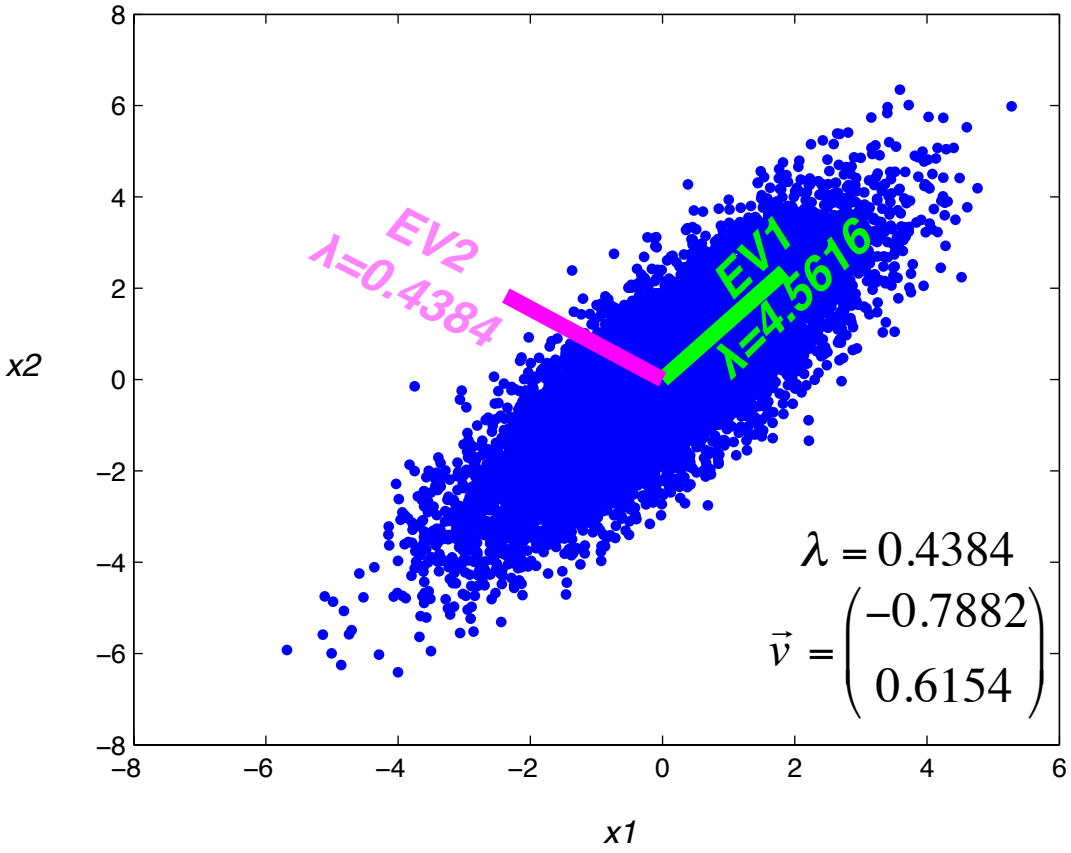


Let's subtract
the mean first
(makes things simple)

PCA – Let's build some intuition

$$\mu = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \qquad \Sigma = \begin{pmatrix} 2 & 2 \\ 2 & 3 \end{pmatrix}$$

It would be nice to
diagonalize the covariance matrix
then you have only think about variance
Think eigenvectors of covariance matrix



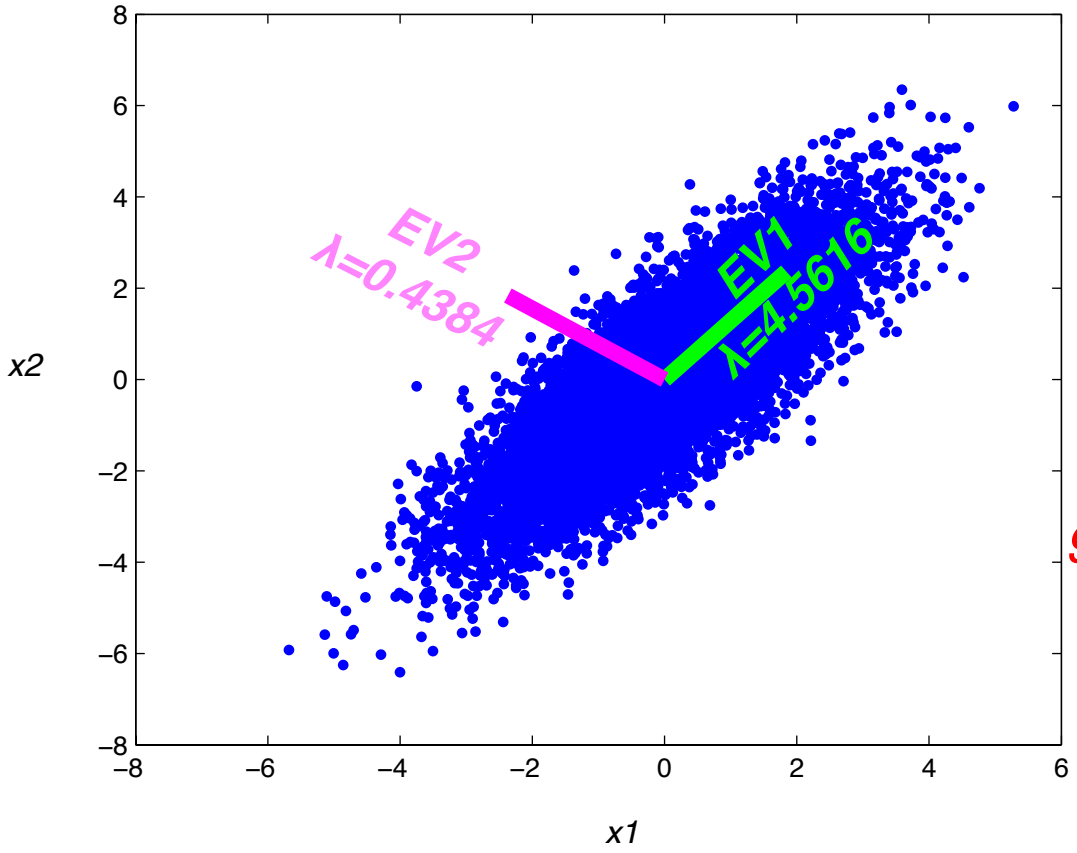
$$\lambda = 0.4384 \qquad \vec{v} = \begin{pmatrix} -0.7882 \\ 0.6154 \end{pmatrix}$$

$$\lambda = 4.5616 \qquad \vec{v} = \begin{pmatrix} 0.6154 \\ 0.7882 \end{pmatrix}$$

PCA – Let's build some intuition

$$\mu = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \qquad \Sigma = \begin{pmatrix} 2 & 2 \\ 2 & 3 \end{pmatrix}$$

It would be nice to
diagonalize the covariance matrix
then you have only think about variance
Think eigenvectors of covariance matrix

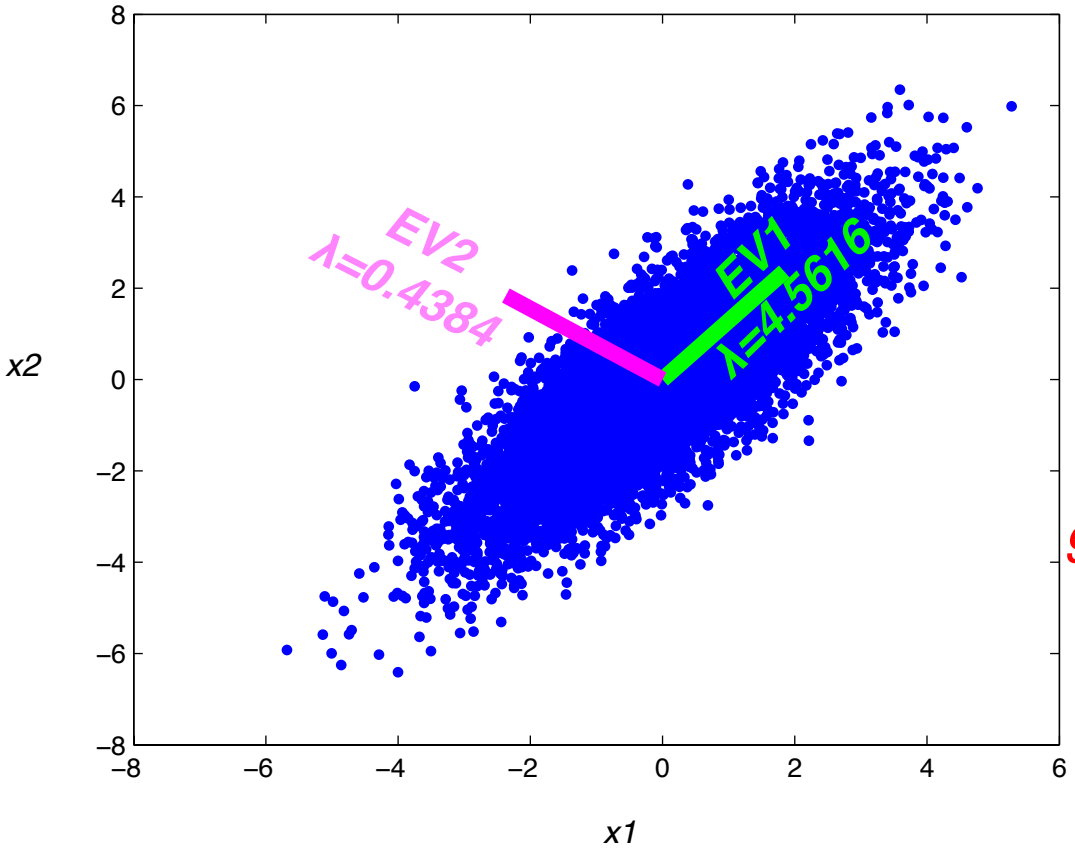


First eigenvector
accounts for
91.23% of the data

PCA – Let's build some intuition

$$\mu = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \qquad \Sigma = \begin{pmatrix} 2 & 2 \\ 2 & 3 \end{pmatrix}$$

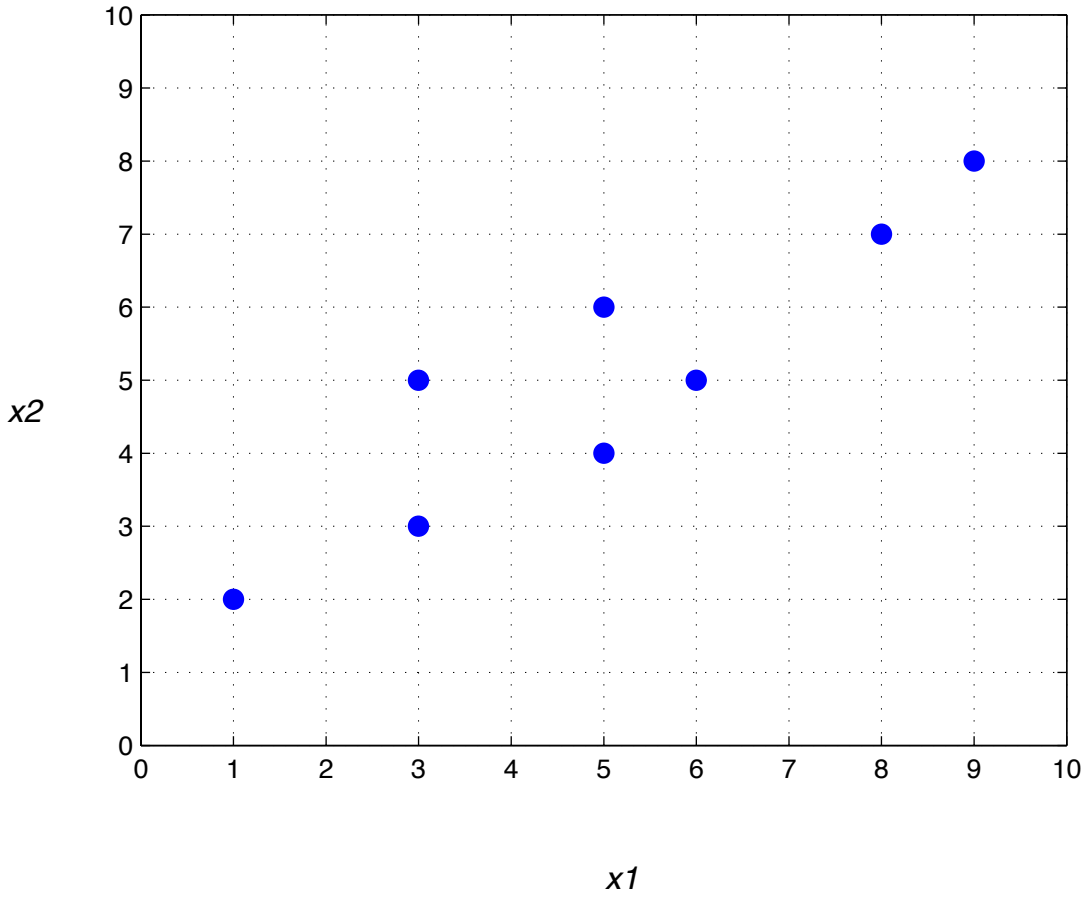
If I have to pick a vector that would approximate this 2-D space?
Green vector right



First eigenvector accounts for 91.23% of the data

PCA – An Example

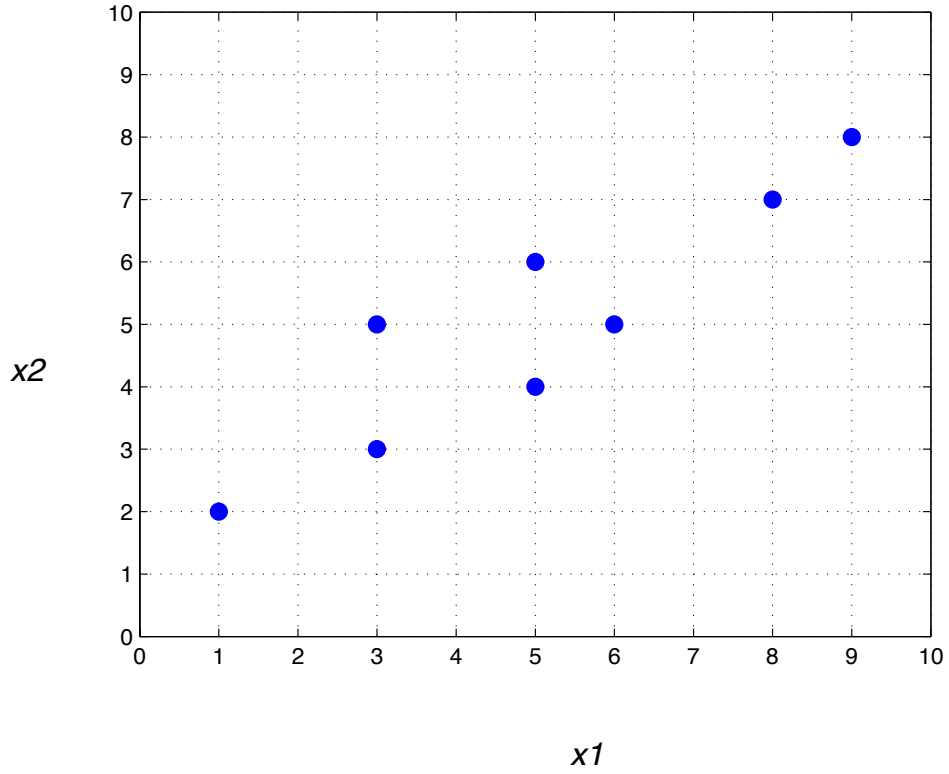
- **Compute the principal components for the following two-dimensional dataset**
  - X=(x1,x2)={(1,2),(3,3),(3,5),(5,4),(5,6),(6,5),(8,7),(9,8)}



PCA – An Example

- **Compute the principal components for the following two-dimensional dataset**
  - X=(x1,x2)={(1,2),(3,3),(3,5),(5,4),(5,6),(6,5),(8,7),(9,8)}

Step 1: Determine the Sample Covariance Matrix



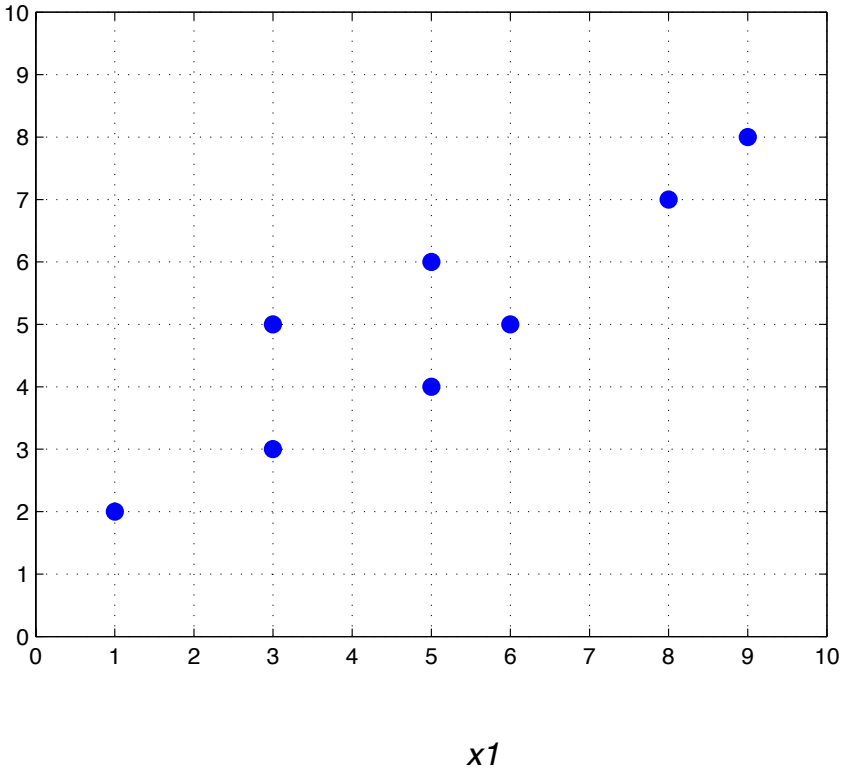
PCA – An Example

- **Compute the principal components for the following two-dimensional dataset**
  - X=(x1,x2)={(1,2),(3,3),(3,5),(5,4),(5,6),(6,5),(8,7),(9,8)}

Step 1: Determine the Sample Covariance Matrix

$$\Sigma = \begin{pmatrix} 6.25 & 4.25 \\ 4.25 & 3.5 \end{pmatrix}$$

Step 2: Find eigenvalues and eigenvectors of the covariance matrix

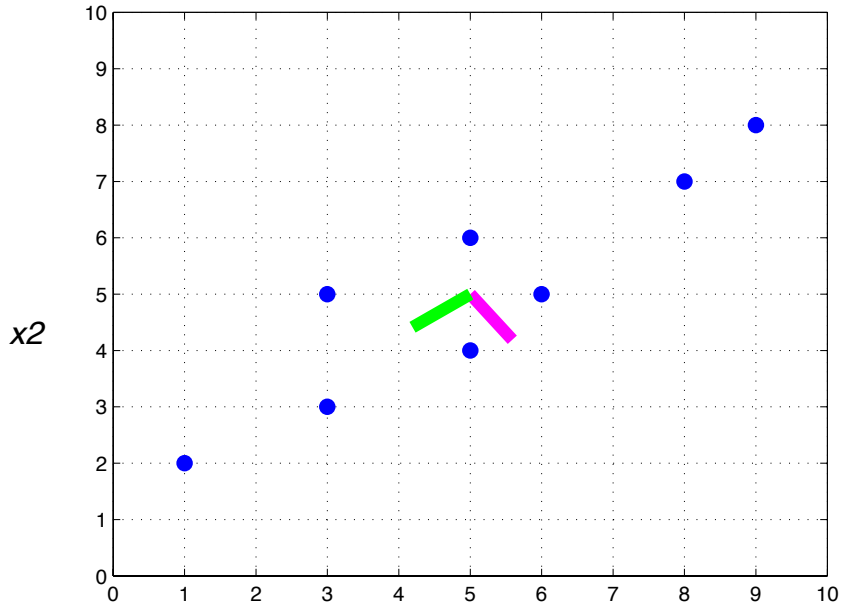


PCA – An Example

- **Compute the principal components for the following two-dimensional dataset**
  - X=(x1,x2)={(1,2),(3,3),(3,5),(5,4),(5,6),(6,5),(8,7),(9,8)}

Step 1: Determine Covariance Matrix

$$\Sigma = \begin{pmatrix} 6.25 & 4.25 \\ 4.25 & 3.5 \end{pmatrix}$$



Step 2: Find eigenvalues and eigenvectors of the covariance matrix

$$\lambda = 0.4081 \qquad \vec{v} = \begin{pmatrix} 0.5883 \\ -0.8086 \end{pmatrix}$$

$$\lambda = 9.3419 \qquad \vec{v} = \begin{pmatrix} -0.5883 \\ -0.8086 \end{pmatrix}$$

Now the Math Part

- **How did we decide that eigenvectors of the covariance matrix will retain maximum information?**

- **What are the assumptions behind PCA**
  - First, the data distribution is unimodal Gaussian in nature
    - Fully explained by the first two moments of the distribution i.e. mean and covariance
  - Information is in the variance

- **PCA dimensionality reduction**
  - The optimal* approximation of a random vector $x \in \Re^N$ by a linear combination of M (M<N) independent vectors is obtained by projecting the random vector x onto the eigenvectors φ_i corresponding to the largest eigenvalues λ_i of the covariance matrix Σ_x

(*optimality is defined as the minimum of the sum-square magnitude of the approximation error)

Lets do the derivation

- **The objective of PCA is to perform dimensionality reduction while preserving as much of the randomness (variance) in the high-dimensional space as possible**
  - Let x be an N-dimensional random vector, represented as a linear combination of orthonormal basis vectors [φ_1l φ_2l ... l φ_N] as

$$\vec{x} = \sum_{i=1}^{N} y_i \varphi_i \qquad where \qquad \varphi_i \perp \varphi_j$$

  - Where y_i is the weighting coefficient of basis vector formed by taking the inner product of x with φ_1:

$$y_i = \vec{x}^T \varphi_i$$

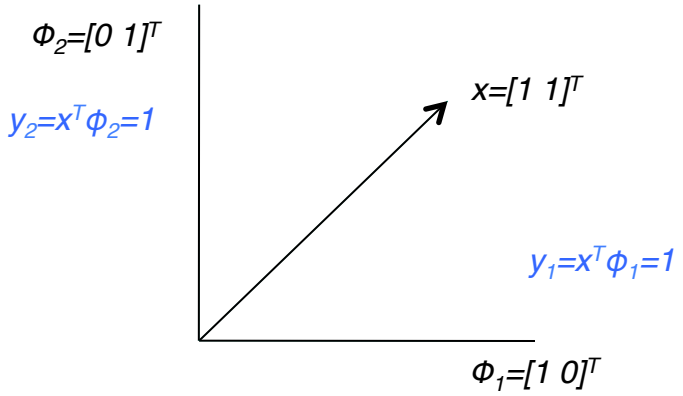
Lets do the derivation

- **The objective of PCA is to perform dimensionality reduction while preserving as much of the randomness (variance) in the high-dimensional space as possible**
  - Let x be an N-dimensional random vector, represented as a linear combination of orthonormal basis vectors [φ_1| φ_2| ... | φ_N] as

$$\vec{x} = \sum_{i=1}^{N} y_i \varphi_i \qquad where \qquad \varphi_i \perp \varphi_j$$

  - Where y_i is the weighting coefficient of basis vector formed by taking the inner product of x with φ_1:

$$y_i = \vec{x}^T \varphi_i$$



$$\vec{x} = y_1 \begin{pmatrix} 1 \\ 0 \end{pmatrix} + y_2 \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Lets do the derivation

- Suppose we choose to represent x with only M (M<N) of the basis vectors. We can do this by replacing the components $[y_{M+1}, \ldots, y_N]^T$ with some pre-selected constants b_i

$$\hat{x}(M) = \sum_{i=1}^{M} y_i \varphi_i + \sum_{i=M+1}^{N} b_i \varphi_i$$

- The representation error is:

Lets do the derivation

- Suppose we choose to represent x with only M (M<N) of the basis vectors. We can do this by replacing the components $[y_{M+1}, \ldots, y_N]^T$ with some pre-selected constants b_i

$$\hat{x}(M) = \sum_{i=1}^{M} y_i \varphi_i + \sum_{i=M+1}^{N} b_i \varphi_i$$

- The representation error is

$$\begin{aligned}
\Delta x(M) &= \vec{x} - \hat{x}(M) \\
&= \sum_{i=1}^{M} y_i \varphi_i + \sum_{i=M+1}^{N} y_i \varphi_i - \left(\sum_{i=1}^{M} y_i \varphi_i + \sum_{i=M+1}^{N} b_i \varphi_i \right) \\
&= \sum_{i=M+1}^{N} y_i \varphi_i - \sum_{i=M+1}^{N} b_i \varphi_i \\
&= \sum_{i=M+1}^{N} (y_i - b_i) \varphi_i
\end{aligned}$$

Lets do the derivation

- **We can measure this representation error by the mean-squared magnitude of Δx**
- **Our goal is to find the basis vectors φ_i and constants *b_i that minimize this mean-square error***

$$E\left[\left|\Delta x(M)\right|^2\right] = \frac{1}{S}\sum_{i=1}^{S}\bar{\varepsilon}_i^{\,2}(M) \qquad S = samples$$

$$= E\left[\left(\sum_{i=M+1}^{N}\left(y_i - b_i\right)\varphi_i\right)\left(\sum_{i=M+1}^{N}\left(y_i - b_i\right)\varphi_i\right)\right]$$

$$= E\left[\left(\sum_{i=M+1}^{N}\sum_{j=M+1}^{N}\left(y_i - b_i\right)\left(y_j - b_j\right)\varphi_i^{\,T}\varphi_j\right)\right]$$

$$= \sum_{i=M+1}^{N} E\left[\left(y_i - b_i\right)^2\right] \qquad \begin{bmatrix}\varphi_i \perp \varphi_j \\ \left|\varphi_i\right| = 1\end{bmatrix}$$

Lets do the derivation

- **Find b_i**
  - The optimal values of b_i *can be found by computing* the partial derivative of the objective function and equating it to zero

$$\frac{\partial}{\partial b_i}\left(\sum_{i=M+1}^{N} E\left[\left(y_i - b_i\right)^2\right]\right) = -2\left(E\left[y_i - b_i\right]\right) = 0$$

$$= -2\left(E\left[y_i\right] - b_i\right) = 0$$

$$b_i = E\left[y_i\right]$$

E[x] is a linear operator
E[A-B]=E[A]-E[B]

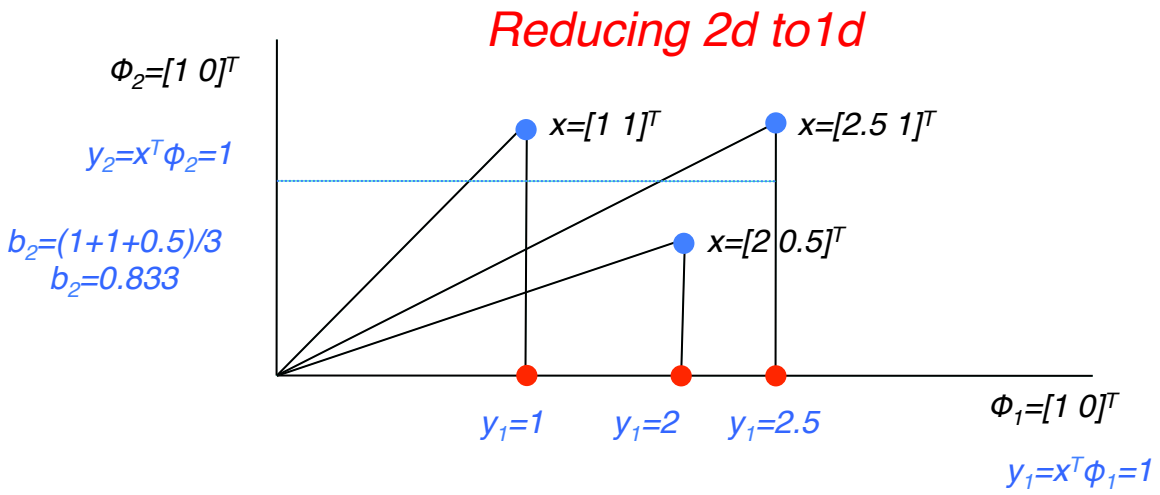
  - Intuitive: replace the discarded dimensions y_i's by their expected value (or mean)

Lets do the derivation

- **Find b_i**
  - As we have done earlier in the course, the optimal values of *bi can be found by computing* the partial derivative of the objective function and equating it to zero

$$b_i = E\left[y_i\right]$$

  - Intuitive: replace the discarded dimensions y_i's by their expected value



Lets do the derivation

- The Mean-Squared-Error can now be written as

$$\bar{\varepsilon}^2(M) = \sum_{i=M+1}^{N} E\left[\left(y_i - b_i\right)^2\right]$$

$$= \sum_{i=M+1}^{N} E\left[\left(y_i - E\left[y_i\right]\right)^2\right]$$

- where

$$y_i = \vec{x}^T \varphi_i$$

Lets do the derivation

- **The Mean-Squared-Error can now be written as**

$$\bar{\varepsilon}^2(M) = \sum_{i=M+1}^{N} E\left[\left(y_i - E\left[y_i\right]\right)^2\right]$$

where

$$y_i = \vec{x}^T \varphi_i$$

- **Substituting y_i in the MSE equation we get:**

$$\bar{\varepsilon}^2(M) = \sum_{i=M+1}^{N} E\left[\left(x^T\varphi_i - E\left[x^T\varphi_i\right]\right)^2\right]$$

$$= \sum_{i=M+1}^{N} \varphi_i^T \underbrace{E\left[\left(x - E\left[x\right]\right)\left(x - E\left[x\right]\right)^T\right]}_{data \quad \text{cov}\, ariance} \varphi_i$$

$$= \sum_{i=M+1}^{N} \varphi_i^T \Sigma_x \varphi_i$$

Where Σ_x is the covariance matrix of x

Lets do the derivation

- We seek to find the solution that minimizes the MSE and is also subject to the normality constraint ($\phi^T\phi=1$), which we incorporate into the expression using a set of Lagrange multipliers λ_i

$$\bar{\varepsilon}^2(M) = \sum_{i=M+1}^{N} \varphi_i^T \Sigma_x \varphi_i + \sum_{i=M+1}^{N} \lambda_i (1 - \varphi_i^T \varphi_i)$$

- Computing partial derivatives with respect to ϕ_i

$$\frac{d}{d\varphi_i}\left(\sum_{i=M+1}^{N} \varphi_i^T \Sigma_x \varphi_i + \sum_{i=M+1}^{N} \lambda_i (1 - \varphi_i^T \varphi_i) \right) = 2\left(\Sigma_x \varphi_i - \lambda_i \varphi_i\right) = 0$$

$$Note: \quad \frac{d}{dx}\left(x^T A x\right) = (A + A^T)x \overset{A \ symmertric}{=} 2Ax$$

$$\Rightarrow \underbrace{\Sigma_x \varphi_i = \lambda_i \varphi_i}_{Eigenvalue \quad problem}$$

- So φ_i and λ_i are the eigenvectors and eigenvalues of the covariance matrix Σ_x

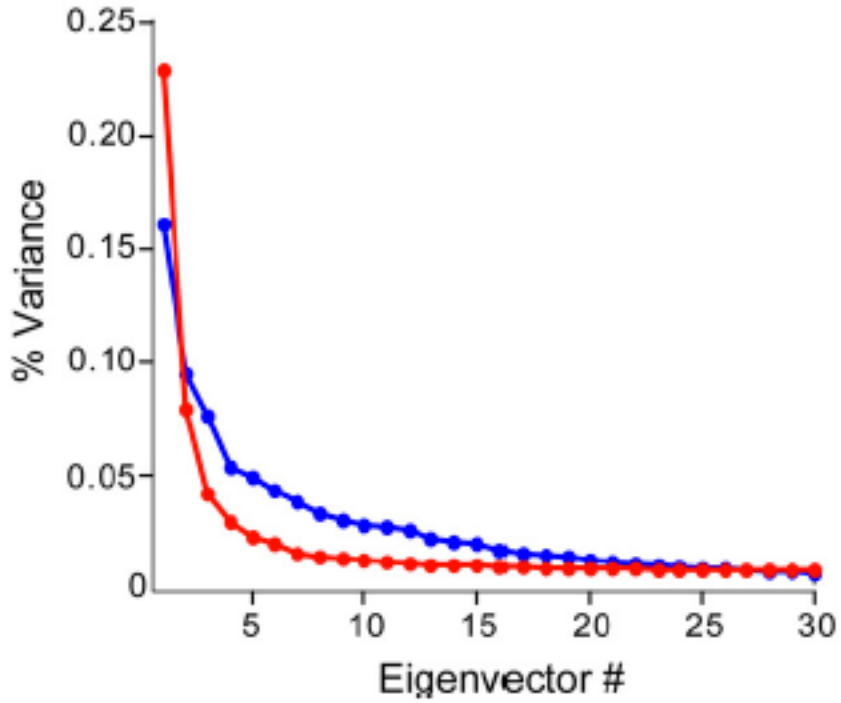
Lets do the derivation

- **We can express the sum-squared error as**

$$\bar{\varepsilon}^2(M) = \sum_{i=M+1}^{N} \varphi_i^T \Sigma_x \varphi_i = \sum_{i=M+1}^{N} \varphi_i^T \lambda_i \varphi_i = \sum_{i=M+1}^{N} \lambda_i$$

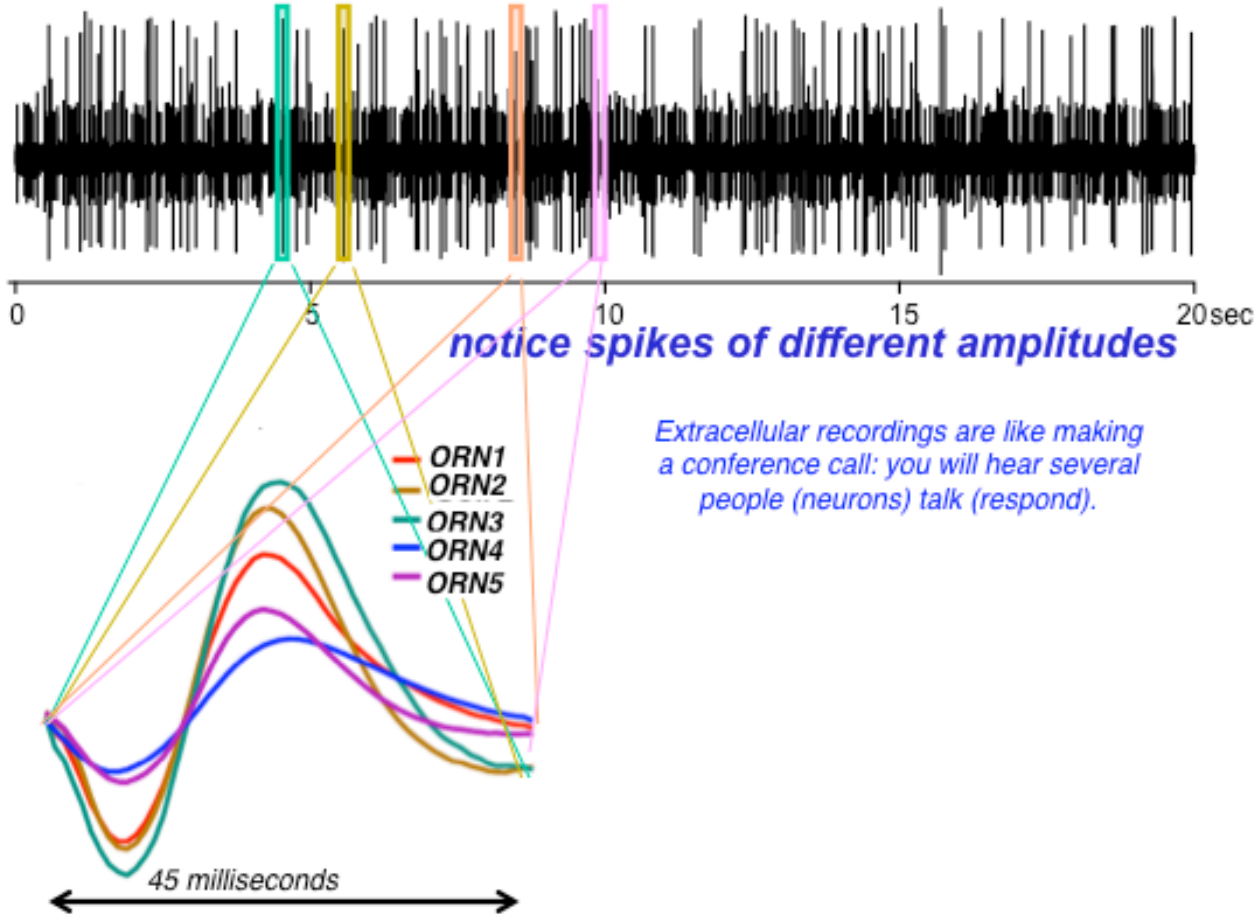
- **In order to minimize this measure, λ_i will have to be smallest eigenvalues**
  - Therefore, to represent *x with minimum sum-square error, we will choose the eigenvectors* φ_i corresponding to the largest eigenvalues λ_i

Scree plot



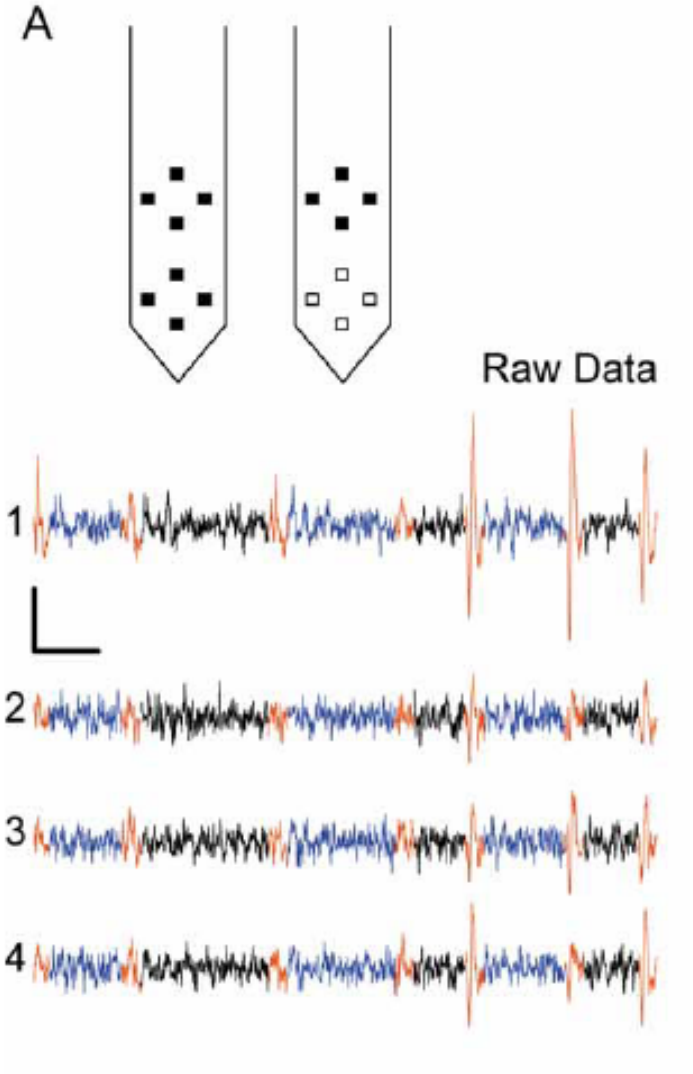
Applications of PCA: Spike Sorting

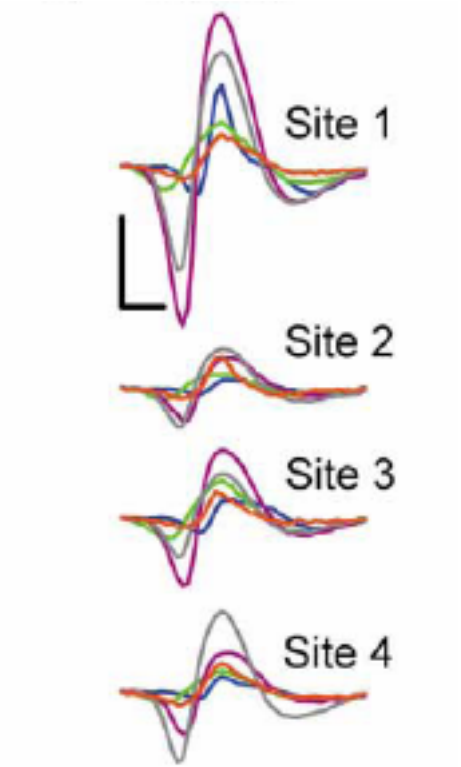
- **Extracellular recording (single electrode)**



Applications of PCA: Spike Sorting

- **Multi-unit recording**

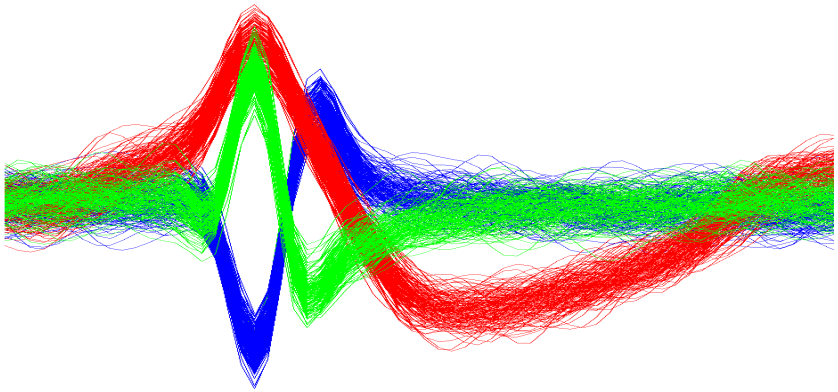




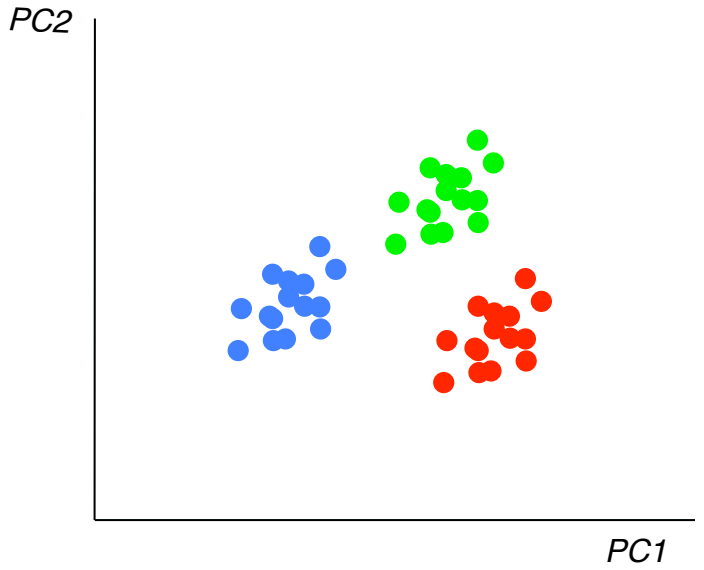
Notice different colors (represent different neurons) have different patterns of extracellular activity) [from Pouzat et al. 2002]

Applications of PCA: Spike Sorting

- **Multi-unit recording**



Different colors (represent different neurons) have different patterns of extracellular activity)



Clustering after PCA allows identification of events from a single source (neuron)

Applications of PCA: Neural Coding

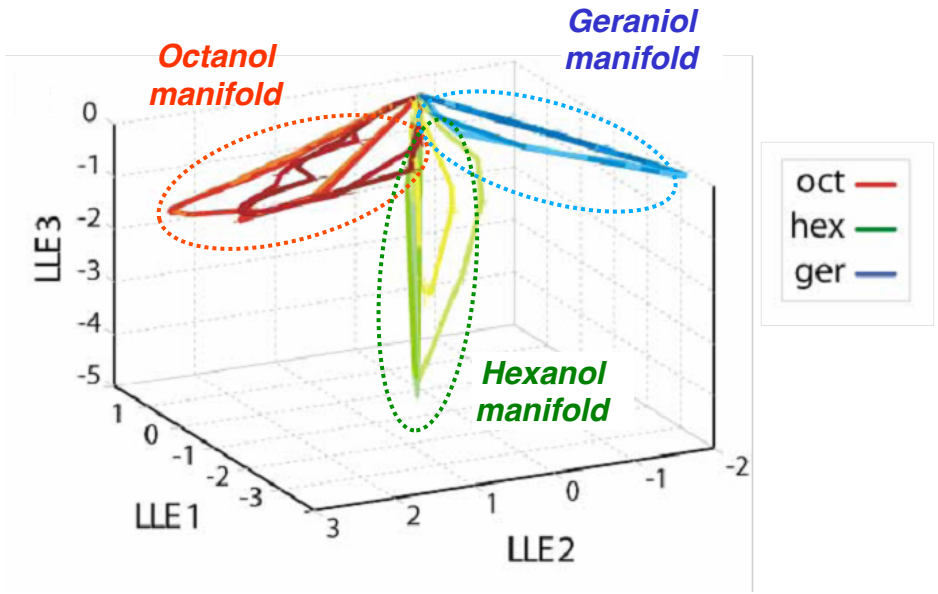
- **Information is represented by spiking patterns of neural ensembles**

- **Visualization of high-dimensional neural activity**

Identity vs. Intensity Coding

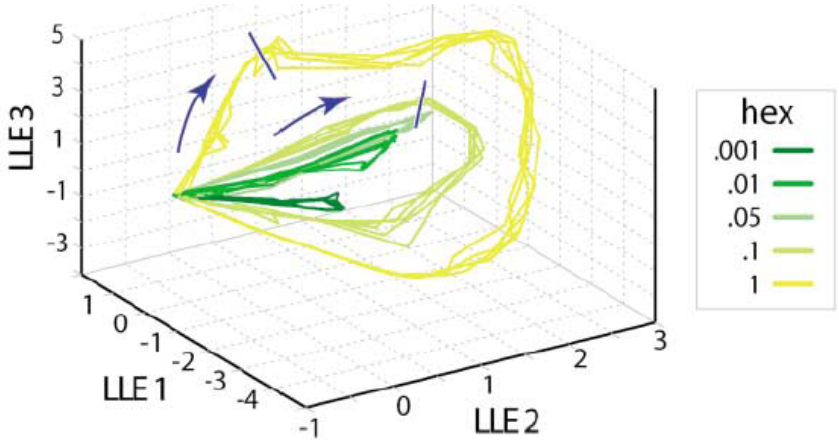
- **PN ensemble activity trace concentration-specific trajectories on odor-specific manifolds (*Stopfer et al., Neuron, 2003*).**

Odor Trajectories
(3 odors at multiple concentrations)



Stopfer et al, Neuron, 2003

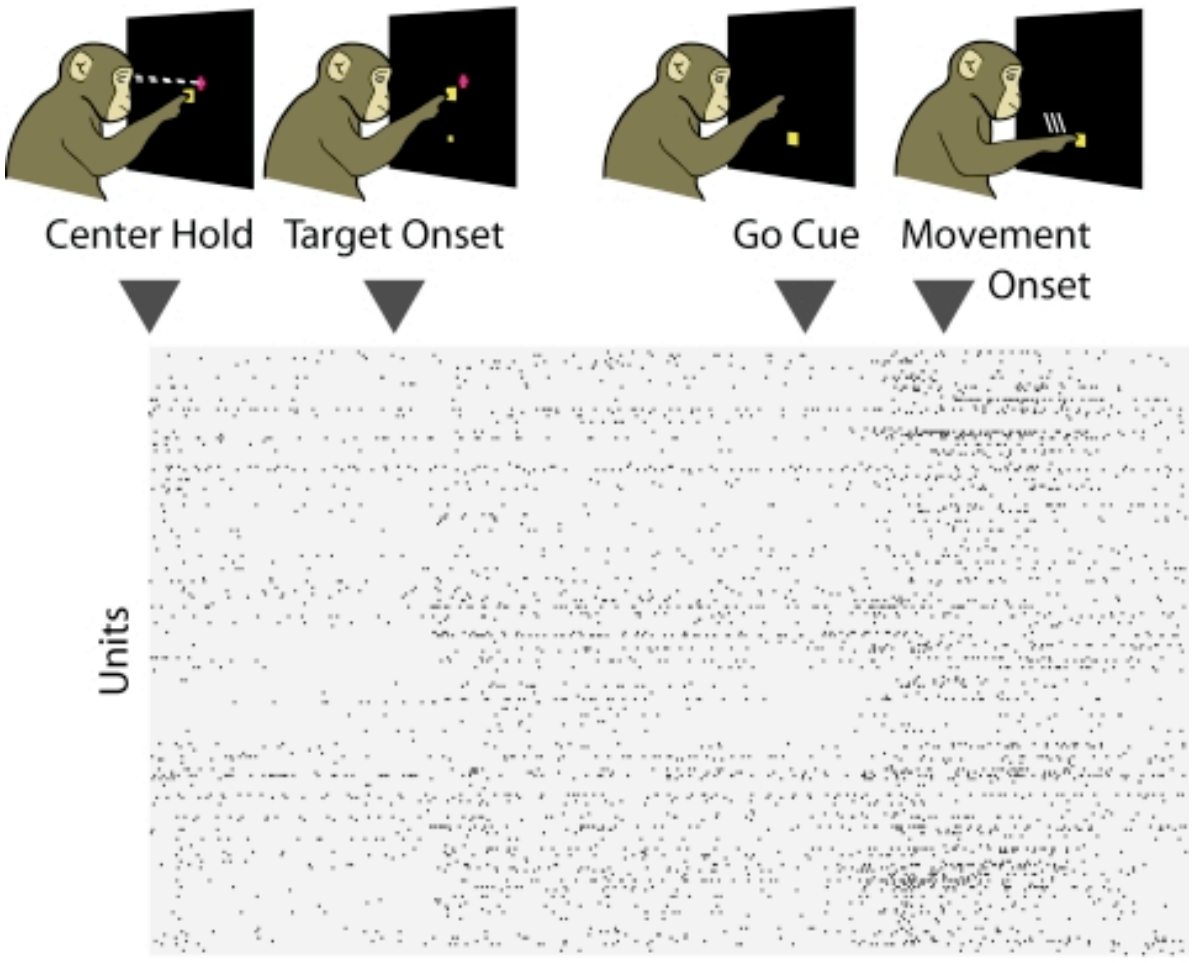
Odor Trajectories
(Hexanol at five concentrations)



Stopfer et al., Neuron, 2003

LLE – Local Linear Embedding
S. Roweis and L. Saul, Science, 2000

Another Neural Coding Example



Kemere et al., 2008
Byron Yu, CMU, Teaching Notes

Another Neural Coding Example

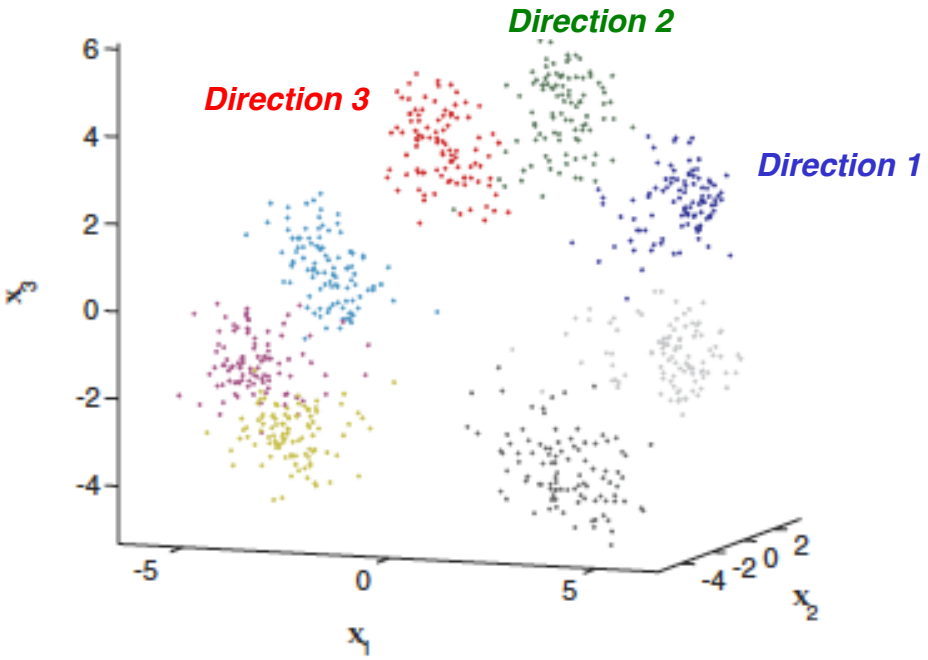
Acquire Touch
Acquire Fix

Target Cue
Delay Period

Go Cue

Movement Period
Target Acquired

Neural representation after dimensionality reduction



Santhanam et al., 2009
Byron Yu, CMU, Teaching Notes

A 2D Numerical Example

PCA Example – Data

- Original data

x	y
2.5	2.4
0.5	0.7
2.2	2.9
1.9	2.2
3.1	3
2.3	2.7
2	1.6
1	1.1
1.5	1.6
1.1	0.9

STEP 1

- Subtract the mean

- from each of the data dimensions. All the x values have average (x) subtracted and y values have average (y) subtracted from them. This produces a data set whose mean is zero.

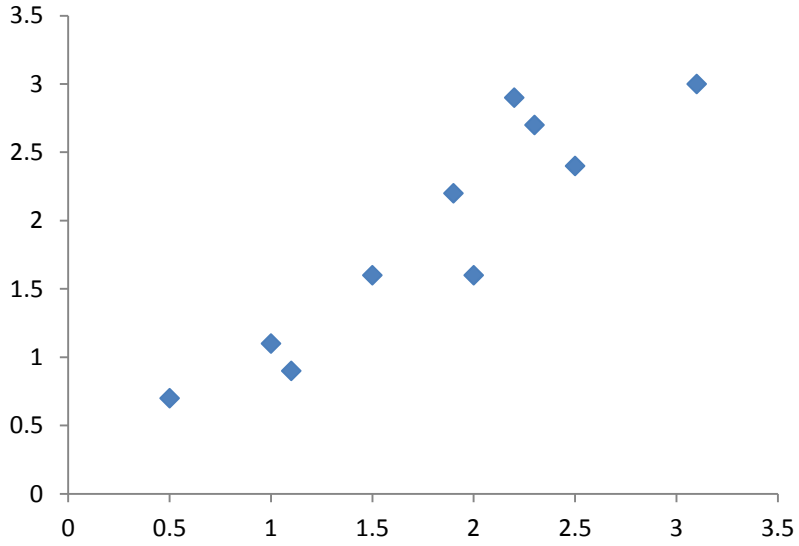
- Subtracting the mean makes variance and covariance calculation easier by simplifying their equations. The variance and co-variance values are not affected by the mean value.

STEP 1

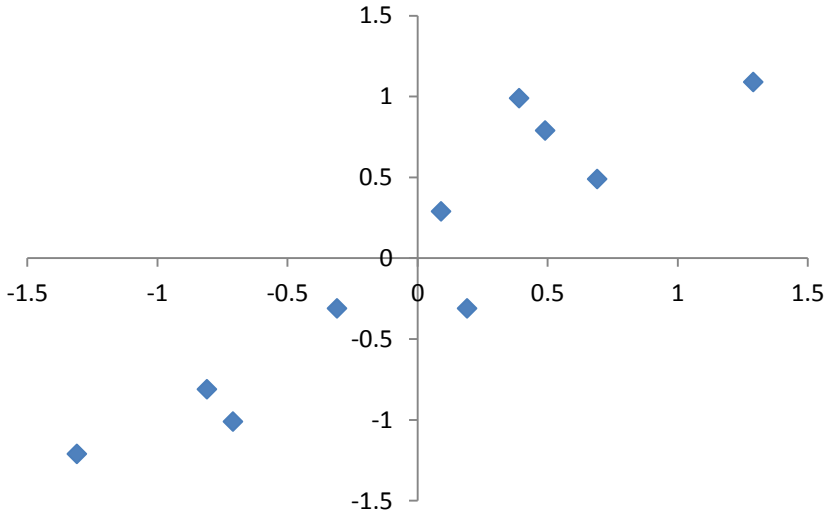
- Zero-mean data

0.69	0.49
-1.31	-1.21
0.39	0.99
0.09	0.29
1.29	1.09
0.49	0.79
0.19	-0.31
-0.81	-0.81
-0.31	-0.31
-0.71	-1.01

STEP 1



Original



Zero-mean

STEP 2

- Calculate the covariance matrix

$$\text{cov} = \begin{pmatrix} .616555556 & .615444444 \\ .615444444 & .716555556 \end{pmatrix}$$

- since the non-diagonal elements in this covariance matrix are positive, we should expect that both the x and y variable increase together.

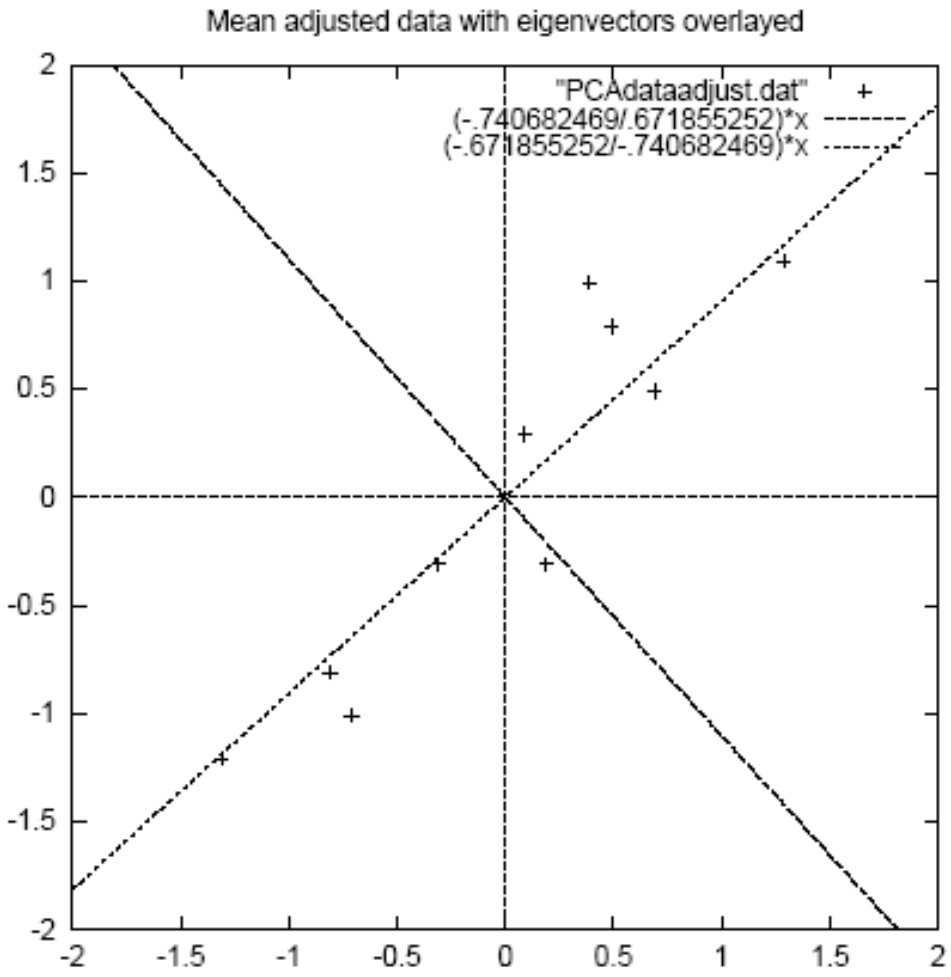
STEP 3

- Calculate the eigenvectors and eigenvalues of the covariance matrix

$$\text{eigenvalues} = \begin{pmatrix} .0490833989 \\ 1.28402771 \end{pmatrix}$$

$$\text{eigenvectors} = \begin{pmatrix} -.735178656 & -.677873399 \\ .677873399 & -.735178656 \end{pmatrix}$$

STEP 3



- eigenvectors are plotted as diagonal dotted lines on the plot.
- Note they are perpendicular to each other.
- Note one of the eigenvectors goes through the middle of the points, like drawing a line of best fit.
- The second eigenvector gives us the other, less important, pattern in the data, that all the points follow the main line, but are off to the side of the main line by some amount.

Figure 3.2: A plot of the normalised data (mean subtracted) with the eigenvectors of the covariance matrix overlayed on top.

Feature Extraction

- Reduce dimensionality and form *feature vector*
  - the eigenvector with the *highest* eigenvalue is the *principal component* of the data set.
  - In our example, the eigenvector with the larges eigenvalue was the one that pointed down the middle of the data.
  - Once eigenvectors are found from the covariance matrix, the next step is to order them by eigenvalue, highest to lowest. This gives you the components in order of significance.

Feature Extraction

- Eigen Feature Vector

  FeatureVector = (eig_1 eig_2 eig_3 … eig_n)

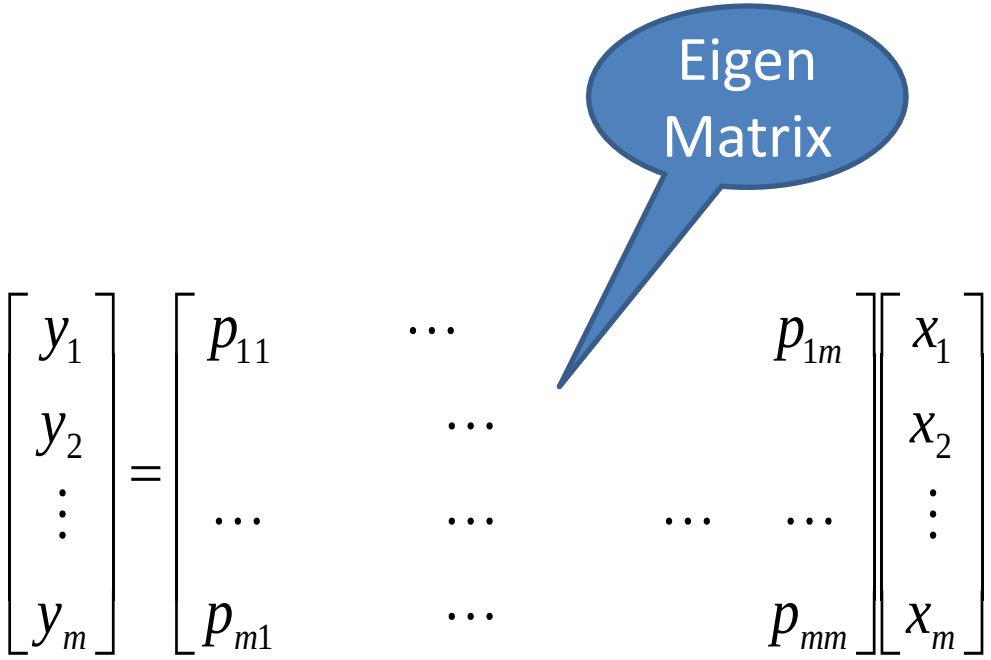
  We can either form a feature vector with both of the eigenvectors:

$$\begin{pmatrix} -.677873399 & -.735178656 \\ -.735178656 & .677873399 \end{pmatrix}$$

  or, we can choose to leave out the smaller, less significant component and only have a single column:

$$\begin{pmatrix} -.677873399 \\ -.735178656 \end{pmatrix}$$

Eigen-analysis/ Karhunen Loeve Transform



Eigen-analysis/ Karhunen Loeve Transform

Back to our example: Transform data to eigen-space (x’ , y’)

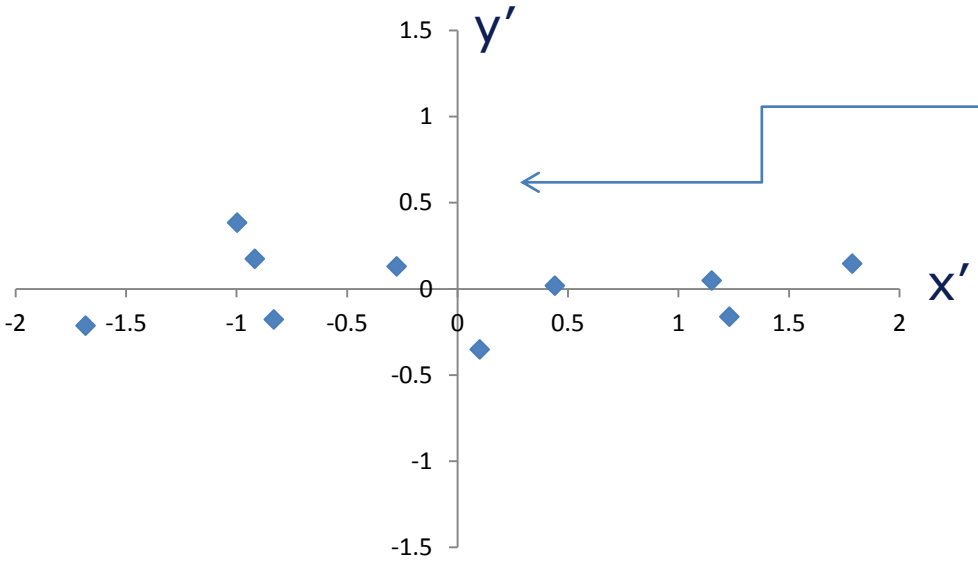
x’ = -0.68x - 0.74y y’ = -0.74x + 0.68y

-.827970186	-.175115307
1.77758033	.142857227
-.992197494	.384374989
-.274210416	.130417207
-1.67580142	-.209498461
-.912949103	.175282444
.0991094375	-.349824698
1.14457216	.0464172582
.438046137	.0177646297
1.22382056	-.162675287

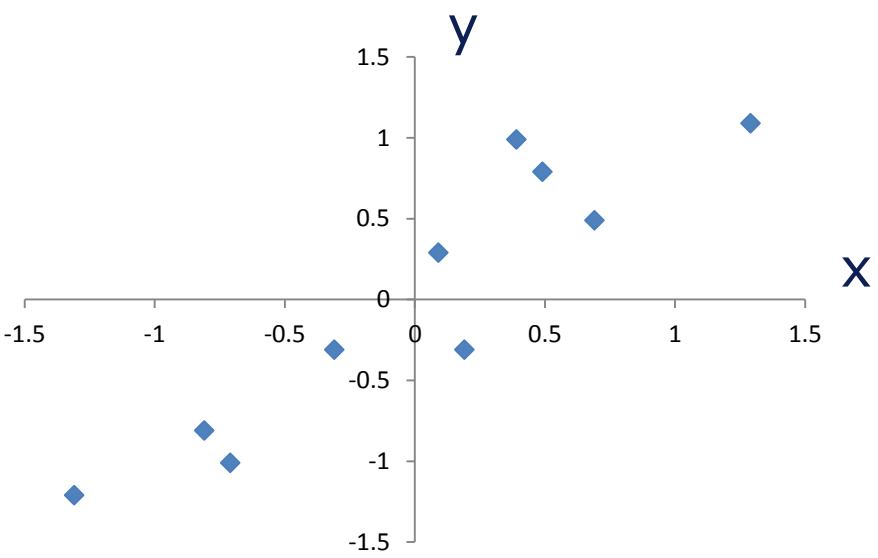
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -0.68 & -0.74 \\ -0.74 & 0.68 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

x	y
0.69	0.49
-1.31	-1.21
0.39	0.99
0.09	0.29
1.29	1.09
0.49	0.79
0.19	-0.31
-0.81	-0.81
-0.31	-0.31
-0.71	-1.01

Eigen-analysis/ Karhunen Loeve Transform



$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -0.68 & -0.74 \\ -0.74 & 0.68 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$



Reconstruction of original Data/Inverse Transformation

- Forward Transform

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -0.68 & -0.74 \\ -0.74 & 0.68 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

- Inverse Transform

$$\begin{bmatrix} x_{construction} \\ y_{construction} \end{bmatrix} = \begin{bmatrix} -0.68 & -0.74 \\ -0.74 & 0.68 \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix}$$

Reconstruction of original Data/Inverse Transformation

- If we reduced the dimensionality, obviously, when reconstructing the data we would lose those dimensions we chose to discard.

- Thrown away the less important one, throw away y' and only keep x'

$$\begin{bmatrix} x_{reconstruction} \\ y_{reconstruction} \end{bmatrix} = \begin{bmatrix} -0.68 \\ -0.74 \end{bmatrix} [x']$$

Reconstruction of original Data/Inverse Transformation

x’

-.827970186

1.77758033

-.992197494

-.274210416

-1.67580142

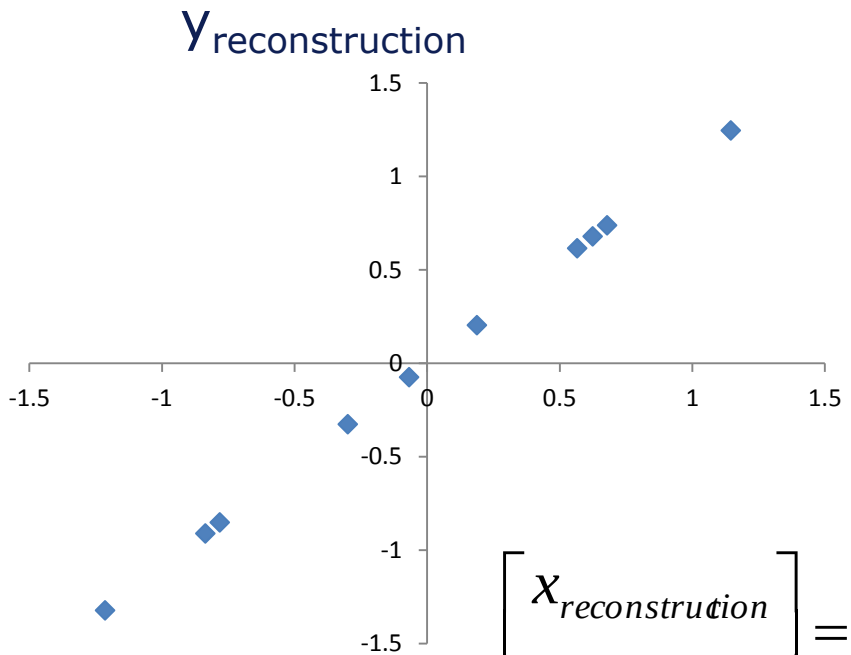
-.912949103

.0991094375

1.14457216

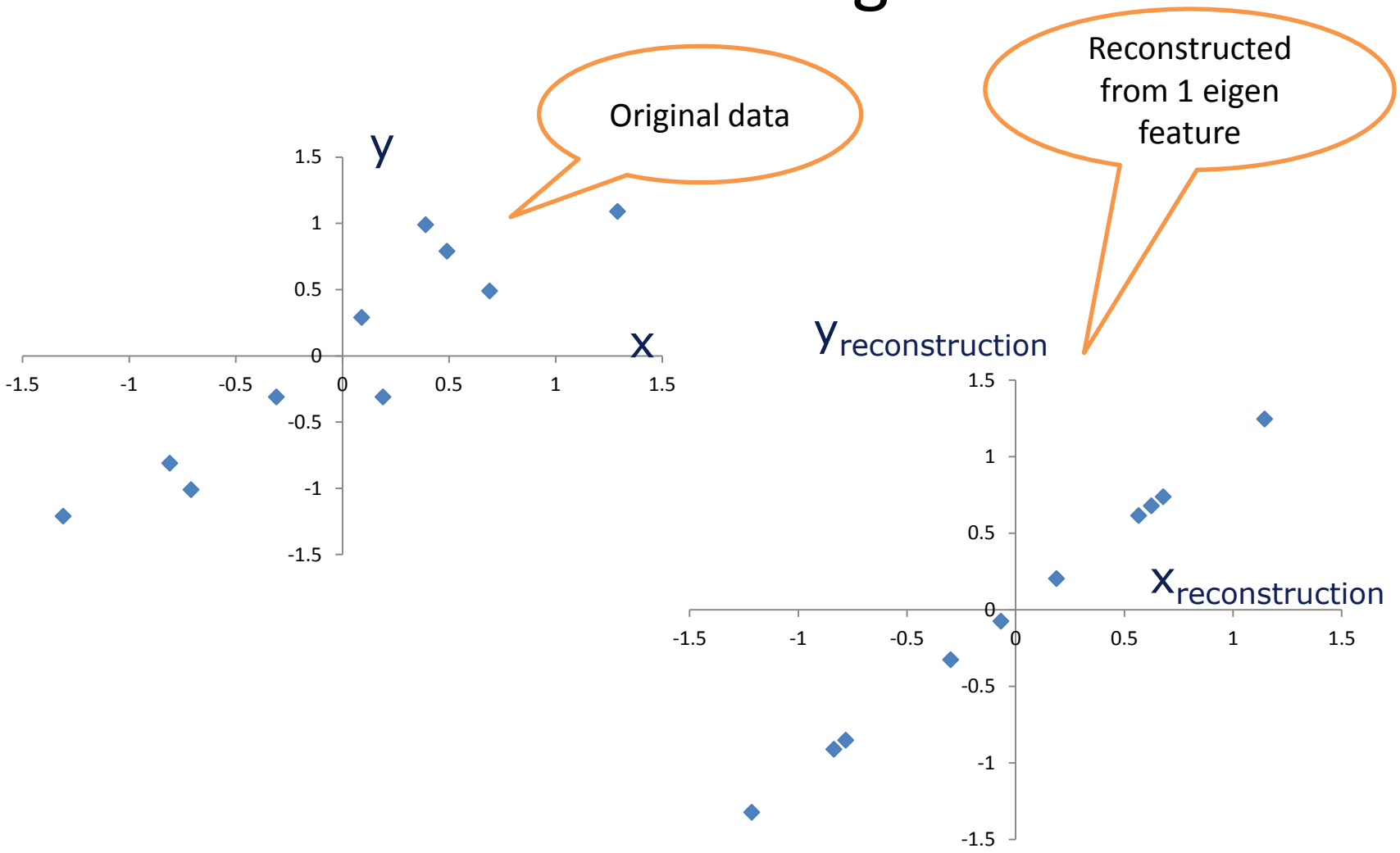
.438046137

1.22382056

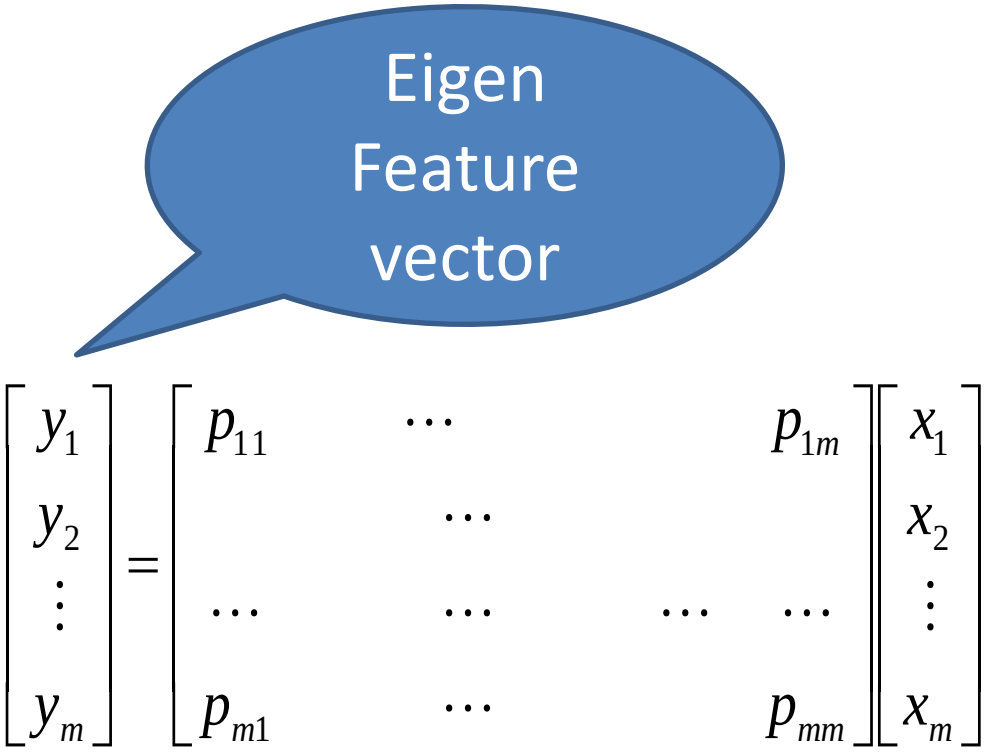


$$\begin{bmatrix} x_{reconstruction} \\ y_{reconstruction} \end{bmatrix} = \begin{bmatrix} -0.68 \\ -0.74 \end{bmatrix} [x']$$

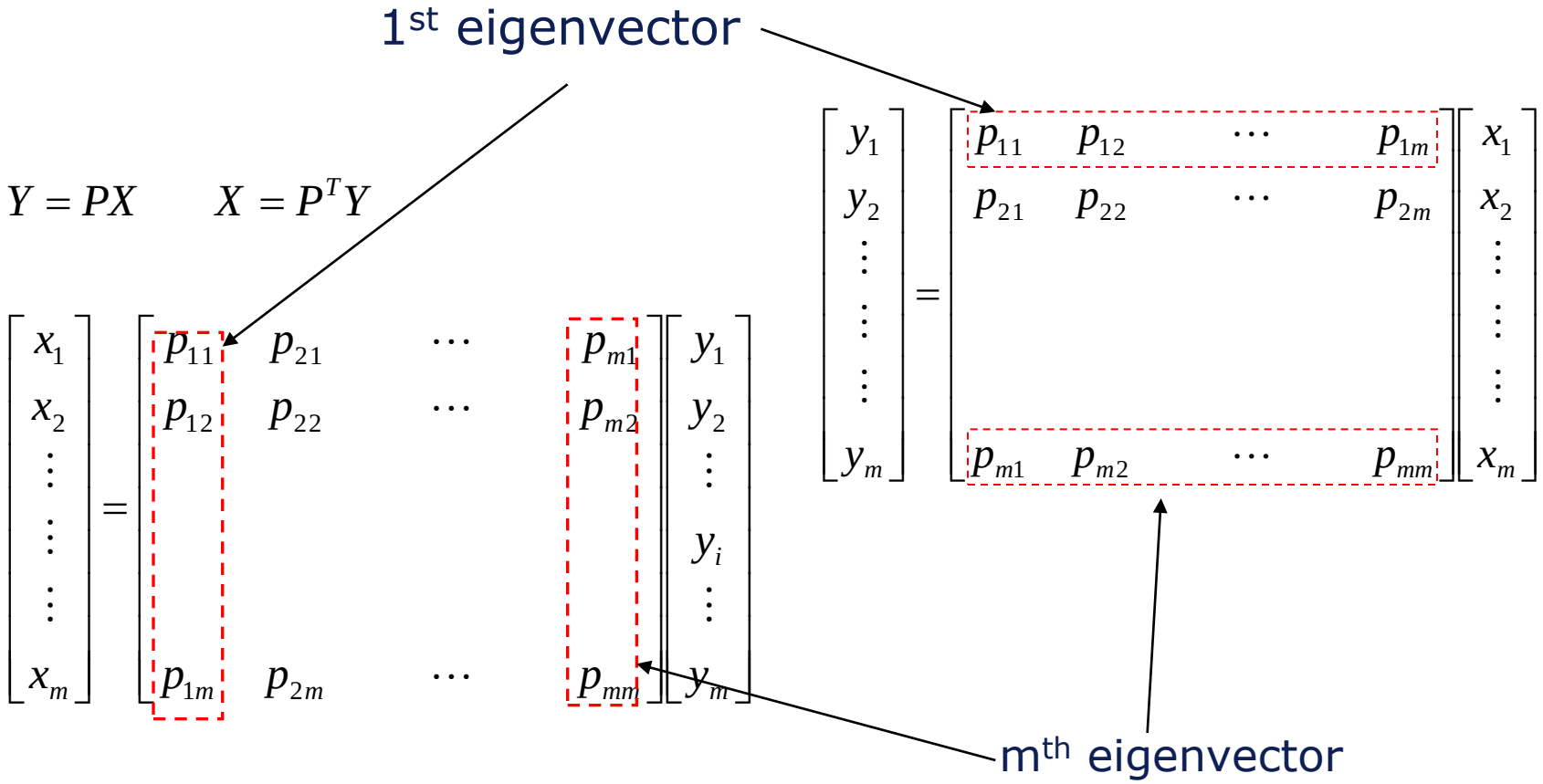
Reconstruction of original Data



Feature Extraction/Eigen-features



PCA Applications –General



- Data compression/dimensionality reduction

PCA Applications -General

- Data compression/dimensionality reduction

$$p_i = \begin{bmatrix} p_{i1} & p_{i2} & \cdots & p_{im} \end{bmatrix}$$

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_i \\ \vdots \\ x_m \end{bmatrix} = \begin{bmatrix} y_1 p_1^T + y_2 p_2^T + \cdots + y_m p_m^T \end{bmatrix}$$

PCA Applications -General

- Data compression/dimensionality reduction
- Reduce the number of features needed for effective data representation by discarding those features having small variances
- The most interesting dynamics occur only in the first *l* dimensions (*l* << m).

$$\hat{X} = \begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \\ \vdots \\ \vdots \\ \hat{x}_m \end{bmatrix} = \begin{bmatrix} p_{11} & p_{21} & p_{l1} \\ p_{12} & p_{22} & p_{l2} \\ & & \\ & & \\ p_{1m} & p_{2m} & p_{lm} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_l \end{bmatrix} = \left[y_1 p_1^T + y_2 p_2^T + \cdots + y_l p_l^T \right]$$

$$p_i = \begin{bmatrix} p_{i1} & p_{i2} & \cdots & p_{im} \end{bmatrix} \qquad X = \left[y_1 p_1^T + y_2 p_2^T + \cdots + y_m p_m^T \right]$$

PCA Applications -General

- Data compression/dimensionality reducti
- Reduce the number of features needed
  by discarding those features having small
- The most interesting dynamics occur only i first *l* dimensions (*l* << m).

We know what can be thrown away; or do we?

$$\hat{X} = \begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \\ \vdots \\ \vdots \\ \hat{x}_m \end{bmatrix} = \begin{bmatrix} p_{11} & p_{21} & p_{l1} \\ p_{12} & p_{22} & p_{l2} \\ & & \\ & & \\ p_{1m} & p_{2m} & p_{lm} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_l \end{bmatrix} = \left[y_1 p_1^T + y_2 p_2^T + \cdots + y_l p_l^T \right]$$

$$p_i = \begin{bmatrix} p_{i1} & p_{i2} & \cdots & p_{im} \end{bmatrix}$$

$$X = \left[y_1 p_1^T + y_2 p_2^T + \cdots + y_m p_m^T \right]$$

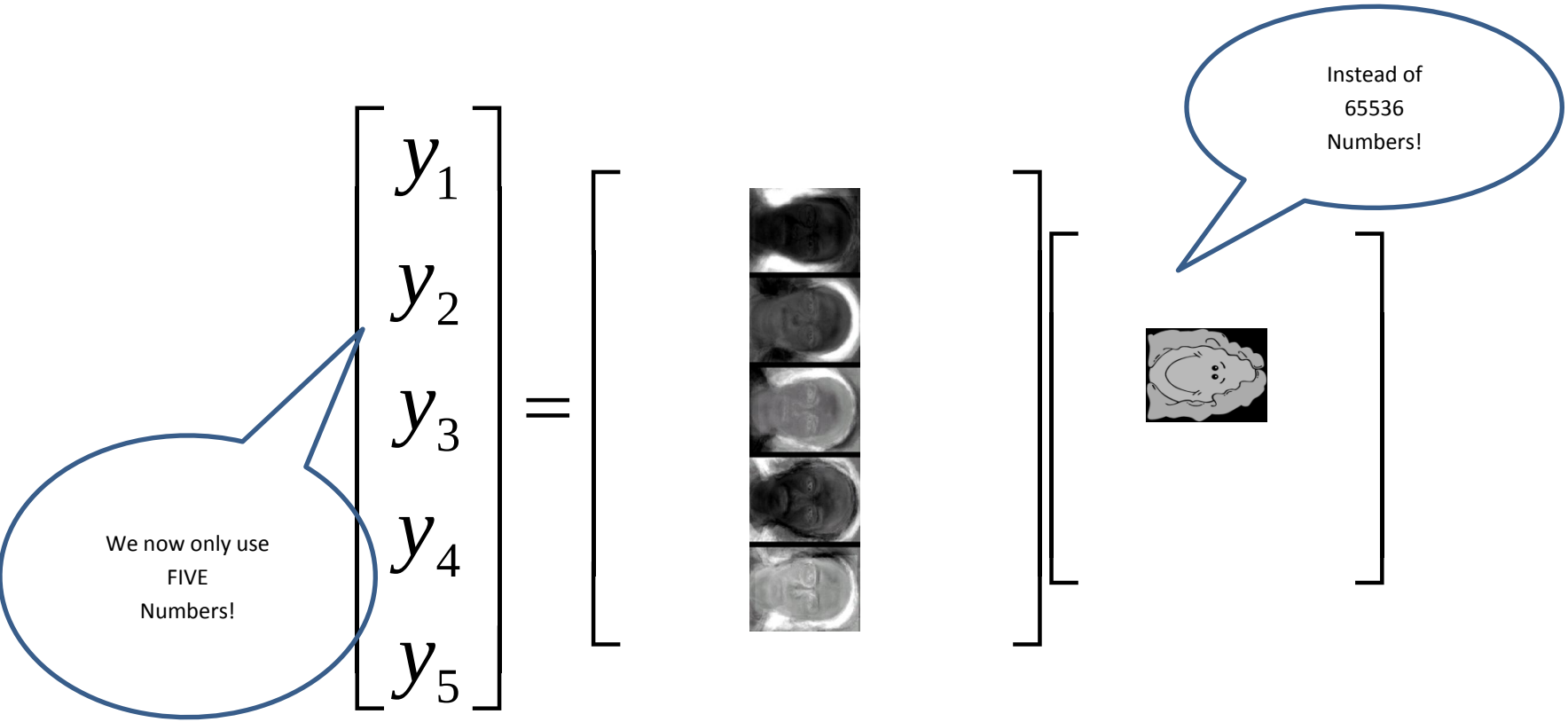
Eigenface Example

- A 256x256 face image, 65536 dimensional vector, X, representing the face images with much lower dimensional vectors for analysis and recognition
  - **Compute the covariance matrix, find its eigenvector and eigenvalue**
  - **Throw away eigenvectors corresponding to small eigenvalues, and keep the first *l* (*l* << m) principal components (eigenvectors)**

p_1 p_2 p_3 p_4 p_5

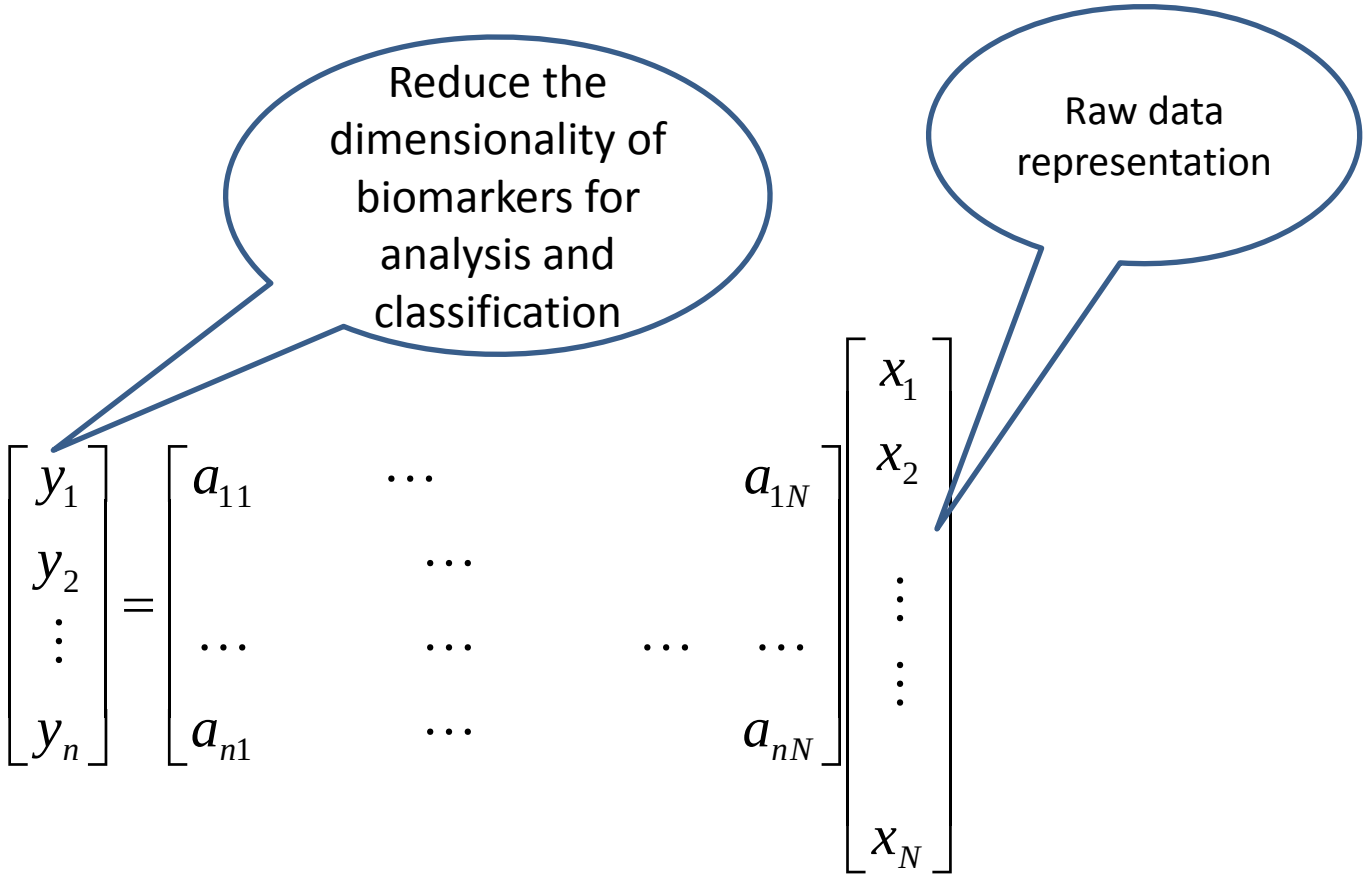
Eigenface Example

- A 256x256 face image, 65536 dimensional vector, X, representing the face images with much lower dimensional vectors for analysis and recognition



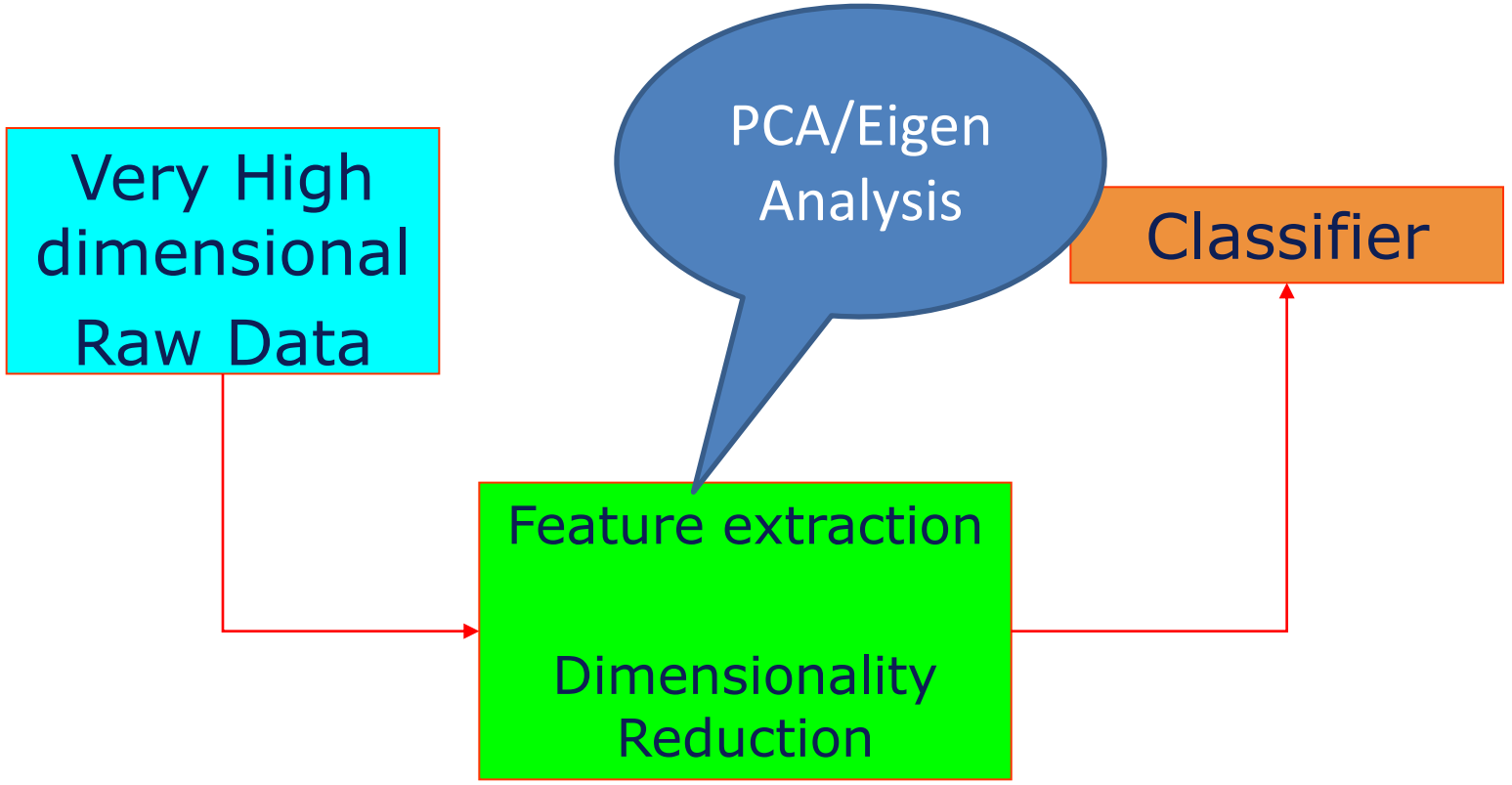
Eigen Analysis - General

– The same principle can be applied to the analysis of many other data types



Processing Methods

- General framework



PCA

- Some remarks about PCA
  - PCA computes projection directions in which variances of the data can be ranked
  - The first few principal components capture the most “energy” or largest variance of the data
  - In classification/recognition tasks, which principal component is more discriminative is unknown

PCA

- Some remarks about PCA
  - Traditional popular practice is to use the first few principal components to represent the original data.
  - However, the subspace spanned by the first few principal components is not necessarily the most discriminative.
  - Therefore, throwing away the principal components with small variances may not be a good idea!